MASTER EN SCIENCES INFORMATIQUES

Les bases de données hétérogènes

Quiévy, Rodrigue

Award date:
1992

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue Grandgagnage, 21
5000 NAMUR

Les Bases de Données Hétérogènes

Rodrigue Quiévy

Mémoire de fin d'études présenté sous la direction du professeur Jean-Luc Hainaut en vue de l'obtention du titre de Licencié et Maître en Informatique

Année académique 1991-1992

R. Quiévy

ERRATA

page 3:1.2, 5ème ligne: lire "lorsqu'on" au lieu de "lorsque l'on"
page 7: 7ème ligne: lire "l'autonomie de ces" au lieu de "l'autonomie des ces"
1.3, 8ème ligne: lire "bases de données" au lieu de bases de donnée"
page 9: 1.4, 18ème ligne: lire "dû" au lieu de "du"
page 11: C, 5ème ligne: lire " répondant à" au lieu de "répondant sa"
page 15: 4ème ligne: lire "amèneront" au lieu de "amènerons"
2.3, 5ème ligne: lire "médiatisées" au lieu de "médiatisée"
7ème ligne: lire "est spécifiée" au lieu de "sont spécifiées"
page 19: 9ème ligne: lire β au lieu de B
page 21: 3ème ligne avant la fin: lire "de la table A" au lieu de "du schéma A"
page 25: 22ème ligne: lire "lorsqu'on" au lieu de "lorsque l'on"
page 42: 7ème ligne lire "de [Motro 87]" au lieu de "par [Motro87]"
page 43: 8ème ligne : lire "du schéma fédéré" au lieu de " de la bases de donnée fédérée"
page 57: 13ème ligne: lire "leur mécanisme" au lieu de "leurs mécanisme"
2ème ligne avant la fin: lire "d'excellents représentants" au lieu de "d'excellent représentant"
page 70: 4.3.1,11ème ligne: lire "transformation de données" au lieu de "transformations de données"
page 71: 7ème ligne: lire "transformation de schémas" au lieu de "transformations de schémas"
page 75: dernière ligne: lire "pour obtenir Q,T,t" au lieu de "pour obtenir Q, t, t^{-1}"
page 76: figure 4.12, lire "règles de mapping T_i,t_i" au lieu de "règles de mapping t_i, t^{-1}_i
C, 2ème ligne: "lire définition" au lieu de "défintion"
page 77: Tableau 4.1, 6ème ligne: lire "élément de la structure P ↵ élément de la structure Q" au lieu de "élément de la structure Q ↵ élément de la structure P"
lire "occurrence" au lieu de occurence" sur toute la page
page 92: figure 4.20, 2ème cadre, colonne de gauche, 5ème ligne: lire" num-cli" au lieu de "um-cli"
page 94: 19ème ligne: lire "nous avons" au lieu de "nousa vons"
24ème ligne: lire "la transformation" au lieu de "le transformation"
page 95: 4.3.4, 1ère ligne: lire" fait" au lieu de "émis fait"
page 99: C, 7ème ligne: lire "sera donnée" au lieu de sera données"
11ème ligne: lire "impossible" au lieu de "impossible impossible"
page 100: D, remplacer les lignes 21 à 25 comme suit:

"- mapping définissant les correspondances entre les éléments du schéma intégré et ceux des schémas exportés
- mapping spécifiant les correspondances entre les occurrences des schémas exportés et ceux du schéma intégré"

page 101 manque (erreur de numérotation des pages)
page 107: B, 3ème ligne: lire "distinguer" au lieu de "distingue"
page 108 manque (erreur de numérotation des pages)
page 131: 2ème ligne: lire "fournit" au lieu de "forunit"
page 153: figure 5.4, lire "rôle(1-1) est-offert -par" au lieu de "rôle (0-1) est- offert -par"
lire "rôle(1-1) est-signé- par" au lieu de "rôle (0-1) est -signé- par"

REMERCIEMENTS

Je tiens tout d'abord à remercier Monsieur Jean-Jacques Snella, Directeur de la Société C-LOG de Genève(Suisse) chez qui j'ai eu le plaisir de faire mon stage de fin d'études.

Je remercie également la Communauté Européenne au travers de COMETT, son projet de formation entre université et entreprise qui m'a permis de financer ce stage de fin d'études.

Je remercie également le Professeur Stephano Spaccapietra et toute son équipe du Laboratoire de Bases de Données de l'Ecole Polytechnique Fédérale de Lausanne avec lesquels j'ai eu de nombreux entretiens lors de mon stage.

Je remercie Monsieur Jean-Luc Hainaut, professeur à l'Institut d'informatique des Facultés Universitaires de Namur et promoteur de ce mémoire, qui m'a guidé dans cette dernière étape par ses remarques, conseils et suggestions.

Je tiens finalement à remercier toute ma famille et mon âme soeur qui m'ont aidées dans cette dernière année d'études en acceptant de leur avoir accordé un peu moins de temps.

Rodrigue Quiévy

Table des matières

CHAPITRE 1: INTRODUCTION

Cette introduction sera tout d'abord consacrée à la description du problème des bases de données hétérogènes. Afin d'être le plus complet possible dans son élaboration nous partirons de la définition des systèmes de bases de données et de leurs composants. Ensuite nous préciserons la place des bases de données hétérogènes dans le cycle de vie d'un système d'information d'une organisation. Cela nous permettra de donner un aperçu historique de l'évolution en matière de stockage des informations, de détecter les origines de ce problème ainsi que les motivations de sa résolution. Pour terminer cette introduction, nous donnerons les différentes philosophies que l'on pourrait adopter face à cette situation.

1.1. Les systèmes de bases de données : définitions et composants

A. Définitions

Un système de base de données (**SBD**) est un système informatique constitué d'un ensemble de données, appelé **base de données,** et d'un logiciel, dénommé **système de gestion de base de données (SGBD)** dont la tâche est de gérer cette base de données. Cet ensemble de données représente une partie du monde réel, généralement nommé **univers de discours**. Les données d'une base de données sont organisées selon un **modèle de données.** La structure effective et l'organisation des données sont décrites par un **schéma de données.** Les utilisateurs accèdent aux données par l'intermédiaire d'une interface, désignée sous le nom **langage de manipulation de données** et fourni par le SGBD. Tout système de gestion de bases de données met également à la disposition de l'utilisateur un **langage de description de données.** Il peut de cette façon transmettre la description des données au SGBD. Un système de base de données doit être implanté sur une **configuration informatique (ordinateur)**. Une personne spécialisée dans les systèmes de bases de données, l'**administrateur de la base de données,** est responsable de la création, de l'évolution et de l'utilisation de la base de données. Tout système de base de données est implanté dans une **organisation** (entreprise, administration,...) dont les **utilisateurs** et **applications** en font usage.
La légende d'icônes présentées à la figure 1.1 sera utilisée tout au long du document afin d'uniformiser la représentation de ces graphiques.

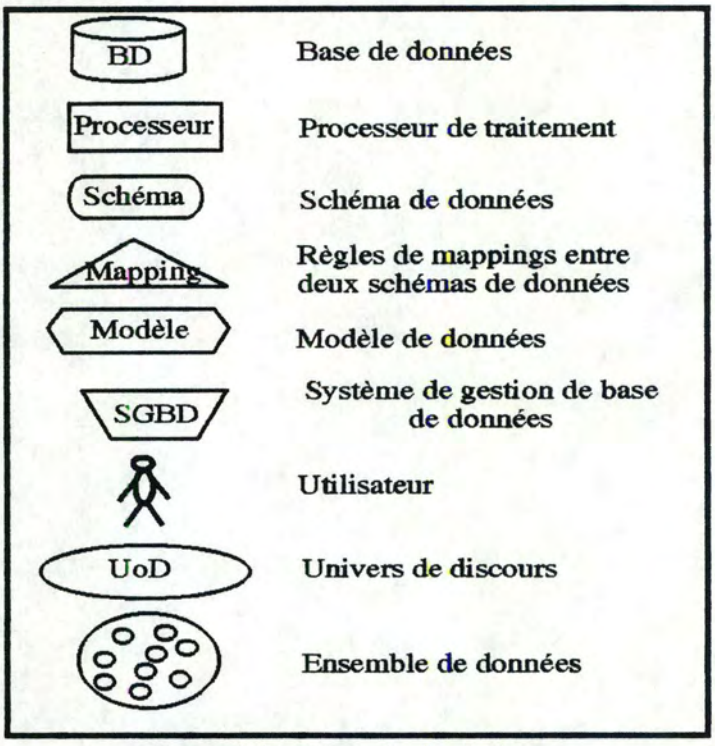


Figure 1.1 : Légende des icônes

La figure 1.2 met en évidence les liens existants entre les composants d'un système de base de données.

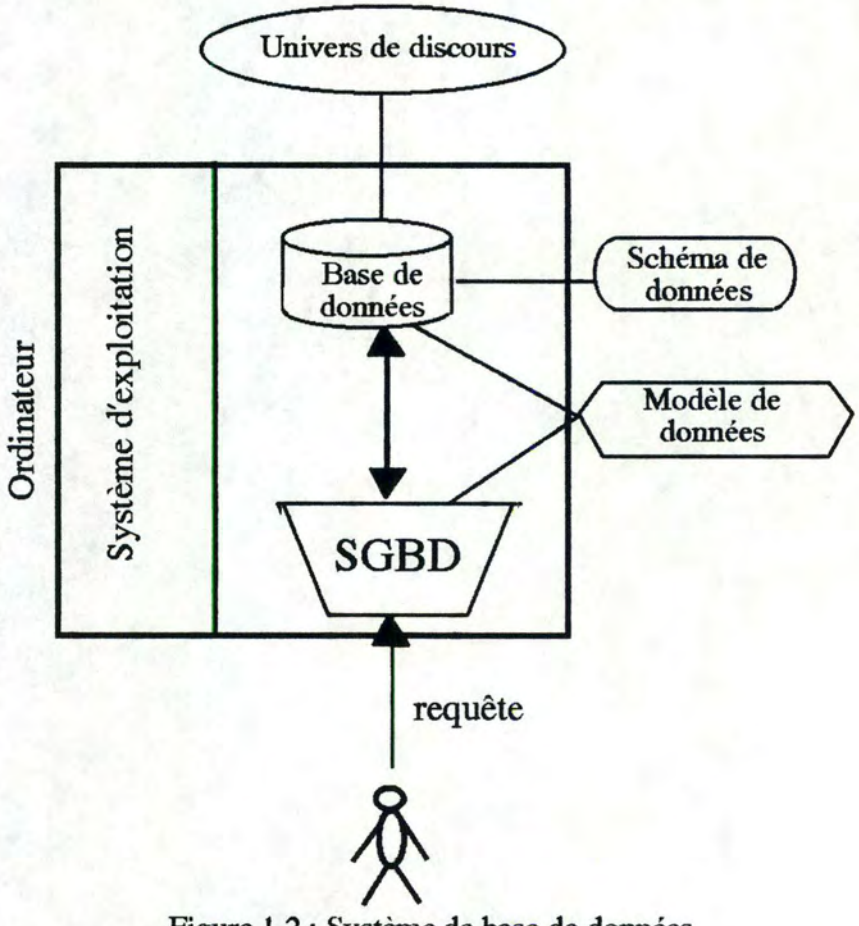


Figure 1.2 : Système de base de données

B. Les composants

Une **base de données** (BD) est une collection de données qui représentent certains aspects statiques et dynamiques d'une organisation et qui sont mises à la disposition des utilisateurs ou applications lorsqu'ils en font la demande. Une base de données doit d'abord constituer une image abstraite fidèle d'un système réel (univers de discours) et ensuite une source de données efficace et fiable.

Un **modèle de données** est un ensemble de concepts qui permettent la description d'une base de données et de règles d'utilisation de ces concepts. Les modèles relationnel, fonctionnel, entité-association, hiérarchique, réseau et orienté-objets sont les grandes catégories de modèles de données actuels.

Un **système de gestion de bases de données** (SGBD) est un logiciel assurant la gestion de la base de données. Il doit offrir toutes une série de fonctionnalités telles que la description et la manipulation de la base de données mais également une gestion performante et fiable de celle-ci. Ces deux dernières caractéristiques nécessitent généralement des techniques complexes (gestion de transaction, reprise après un incident,...). Un SGBD traditionnel assure la gestion d'une base de données conforme à **un et un seul** modèle de données, ne permet l'utilisation que d'un langage de manipulation de données (LMD) et d'un langage de description de donnée (LDD). On parle de SGBD **mono-modèle** et **mono-langage**.

Le **schéma** d'une base de données est l'expression de la description d'une base de données en employant un modèle de données.

1.2. Les systèmes constitués de plusieurs systèmes de base de données

Ce type de système se définit comme un système composé d'un ensemble de SBD et peut être caractérisé selon trois dimensions orthogonales qui vont permettre une classification et une définition de ces différents systèmes. Pour faciliter la compréhension, on parlera de **système global** pour désigner l'entièreté des systèmes et de **système local** lorsqu'on l'on spécifie un système composant le système global.

A. Distribution

Les données peuvent être réparties sur plusieurs bases de données. Celles-ci peuvent être stockées sur un seul ordinateur ou sur plusieurs ordinateurs. Dans ce derniers cas, ces ordinateurs peuvent être localisés sur un même site ou sur des sites différents. Il existe diverses techniques de répartition des données mais elles ne seront pas exposées.
En ce qui concerne la distribution, on peut distinguer deux classes de bases de données : les BD distribuées et les BD non distribuées ou centralisées.

B. Hétérogénéité

A partir des différents composants d'un système de base de données, on peut définir une classification des types d'hétérogénéité que l'on peut rencontrer lors de la comparaison de deux systèmes de bases de données.

- Hétérogénéité des configurations informatiques des systèmes de bases de données

Deux ordinateurs peuvent se distinguer sur base du système d'exploitation utilisé (DOS, UNIX, VMS, OS2,...), des composants matériels (processeur, bus, mémoire,...) mais également des systèmes de communication (types de réseau,...). Ce type d'hétérogénéité concerne fondamentalement l'interconnexion de machines hétérogènes, sujet de recherche important à l'heure actuelle.

- Hétérogénéité des SGBD encore appelée hétérogénéité syntaxique

Trois aspects distinguent les SGBD entre eux : le **modèle de données**, les **langages** et les **caractéristiques techniques.**

Les SGBD offrent des **structures distinctes de représentations de données**. Le tableau 1.1 fait état des concepts principaux utilisés dans différents modèles (relationnel, hiérarchique,...).

Tableau 1.1

Modèle	Concepts	SGBD	Langage LMD
Relationnel	Table, relation, tuple	ORACLE, INGRES	SQL, QUEL
Hiérarchique	Record, champ, segment	IMS	DL/I
Réseau	Article, set, attribut	IDMS	Codasyl DML
Fonctionnel	Entités, fonctions	CAA	DAPLEX
Entité-Association	Type d'entité, d'association, attributs		
Orienté objet	Classe, objet, méthode	O2, ONTOS	

Au niveau des **langages de manipulation de données** des SGBD, on peut trouver des langages assertionnel (Ex:SQL), navigationnel (Ex:Codasyl DML) et graphique. Deux SGBD utilisant le même modèle de données peuvent néanmoins utiliser des langages différents. (Exemple: SQL et QUEL pour le modèle relationnel)

Chaque SGBD utilise des **techniques** différentes pour la gestion des données et notamment pour le traitement des transactions, des reprises après incidents,...

- *Hétérogénéité sémantique*

Cette hétérogénéité relève du schéma de données de la base de données. Deux schémas, décrits dans un même modèle de données, peuvent en effet représenter différemment des concepts identiques du monde réel. Il n'est toutefois pas aisé de fournir une taxonomie précise des conflits donnant naissance à ce type d'hétérogénéité. Celle fournie est issue de [SPACCAPIETRA 91].

Conflit de description: ce type de conflit est observé quand des classes identiques d'objets sont décrites par des ensembles différents de propriétés. Ce sont les conflits de noms (homonymie, synonymie), les différences d'échelle, les différences de domaines...

Conflit sémantique: ce type de conflit est observé quand deux schémas adoptent des classifications différentes pour les mêmes ensembles d'objets. Cela débouche généralement sur des ensembles d'objets qui se recouvrent. Citons par exemple la classe PERSONNE dans le premier schéma et les classes HOMME, FEMME, ENFANT dans le second schéma.

Conflit structurel: ce type de conflit est observé lorsque deux sous-ensembles d'objets identiques du monde réel sont représentés par des structures différentes. En termes entité-association, un sous-ensemble du premier schéma est modélisé par un type d'entité tandis que le sous-ensemble correspondant du second schéma est modélisé par un type d'association.

La pratique montre que la détection de ces conflits est bien plus complexe que leurs résolutions.

- *Hétérogénéité des données*

Les inconsistances de données entre bases de données dont leurs univers de discours se recoupent posent de nombreux problèmes sur le choix de la donnée à fournir à l'utilisateur.
Deux origines à cette inconsistance de données :

- erreur d'introduction des données
- obsolescence d'une donnée due à un retard de mise-à-jour

Citons l'exemple suivant : BD 1 (nom, prénom, domicile): Dupont Henri Namur
BD 2 (nom, prénom, domicile): Dupont Henri Bruxelles
Le domicile de Dupont Henri, est-ce Namur ou Bruxelles ?

On a ainsi exposé les divers types d'hétérogénéité sur base des composants d'un système

de base de données. La complexité du problème de l'hétérogénéité croît du fait du cumul possible de ces quatre types d'hétérogénéité. On peut en effet avoir des situations présentant une quadruple hétérogénéité.

Afin de faciliter la classification des systèmes, l'on parlera de systèmes soit hétérogènes, soit homogènes

C. Autonomie

Le caractère d'autonomie d'un système local vis-à-vis du système global reflète le contrôle dont dispose l'organisation sur ce système local vis-à-vis des autres organisations qui peuvent y accéder. Plus concrètement, le degré d'autonomie se calcule en fonction de deux éléments: le **droit du système local de contrôler l'accès (en consultation et en modification) à ses données par les utilisateurs du système global** et **le droit du système local à accéder et à manipuler ses propres données indépendamment du système global.** Les raisons d'existence de ces droits sont la **performance**, la **sécurité** et l'**accès local sans interférence externe** c'est-à-dire des utilisateurs du système global.

Quatre types d'autonomie sont à observer :

- *Autonomie de conception*

Cette autonomie est respectée si l'organisation propriétaire du système de base de données peut choisir l'univers de discours, le modèle de données, l'interprétation sémantique du schéma, les fonctions du système, le type d'implémentation,...
Ce type d'autonomie est source d'hétérogénéité comme elle a été définie auparavant.

- *Autonomie de communication*

Elle représente la capacité de choisir les systèmes ou organisations avec lesquels communiquer.

- *Autonomie d'exécution*

Elle exprime le choix du système face à l'exécution d'une requête locale ou d'une requête globale. La différence entre ces deux types de requêtes est que la première est issue d'un utilisateur du système local tandis que l'autre est posée par un utilisateur du système global.

- *Autonomie d'association*

Elle exprime la capacité de décision du système vis-à-vis du partage de ses ressources et

opérations.

On peut ainsi classifier les systèmes de bases de données en trois classes : les systèmes **autonomes, semi-autonomes** et **fortement intégrés**. Les systèmes autonomes ne connaissent pas l'existence des autres systèmes. Les systèmes semi-autonomes peuvent opérer indépendamment mais permettent le partage de leurs données. Les systèmes fortement intégrés n'offrent pas d'autonomie locale. Les chapitres suivants nous montreront la **tension** existante entre l'intégration des BD locales et l'autonomie des ces systèmes.

Sur base de ces trois caractéristiques, on peut définir le problème des bases de données hétérogènes.

1.3. Exposé de la situation des Bases de Données Hétérogènes (BDH)

On est face à une collection de bases de données hétérogènes, autonomes et éventuellement distribuée et dont les univers de discours peuvent contenir des éléments communs.
Le défi qui nous est posé consiste à permettre à un utilisateur d'accéder de manière "***transparente***" à cette collection de bases de données. Avant de préciser les conditions sous-jacentes à cette caractéristique qui est imposée à la méthode d'accès, examinons tout d'abord ce que cache le terme "accès transparent".

Par "***accès transparent***", on entend que l'utilisateur pourra accéder à cette collection de bases de donnée avec l'illusion de ne disposer que d'un seul système de base de données. Ainsi l'utilisateur ne voit qu'un schéma de données, n'utilise qu'un seul langage de manipulation et un seul modèle de données. Il ne devra donc pas connaître différents langages de manipulation de données pour accéder à celle-ci, ni différents modèles de données, ni leur localisation physique. Cette transparence sous-entend que le résultat d'une requête soit intégré, complet et non redondant. On n'acceptera ni bruit, ni silence dans le résultat d'une requête formulée par l'utilisateur. De plus la transparence de la distribution et l'autonomie des bases de données doivent être respectées, ce qui signifie qu'il doit toujours être possible de manipuler les bases de données localement.

La solution proposée devra donc offrir une triple transparence : **transparence d'hétérogénéité, de distribution et d'autonomie.**

La figure 1.3 expose la situation initiale et le résultat escompté ainsi que la terminologie qui sera utilisée ultérieurement.

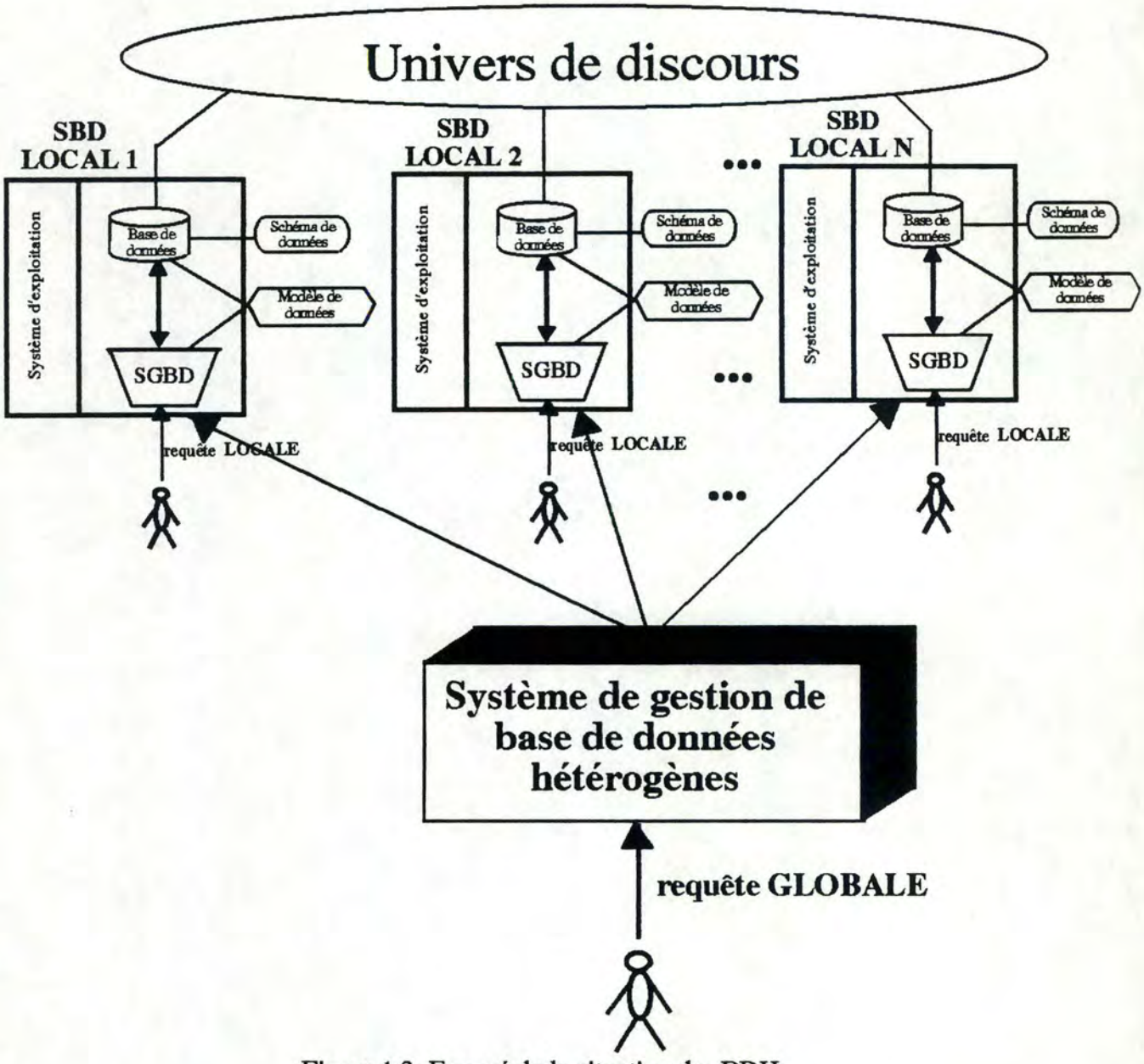


Figure 1.3: Exposé de la situation des BDH

Sur cette figure, on peut voir la distinction entre les différents systèmes de bases de données locaux. Le système de gestion de bases de données hétérogènes (**SGBDH**) aura dont pour tâche de gérer cette collection de bases de données en offrant toutes les fonctionnalités d'un SGBD traditionnel c'est-à-dire la manipulation de données, la gestion des transactions, le contrôle de concurrence, etc... mais également des fonctionnalités qui lui seront particulières pour assurer la gestion d'un tel système.

Sur ce schéma on fait la distinction entre **requête locale** et **requête globale**; une requête locale étant une requête qui est adressée à un SBD local, en dehors du contrôle du SGBDH et qui respecte donc les contraintes du modèle, du schéma et du langage de ce système local. A l'opposé, une requête globale est adressée au système de gestion de bases de données hétérogènes qui est seul habilité à résoudre cette requête. Comme nous le verrons par la suite,

cette requête globale sera résolue à l'aide d'un ensemble de requêtes locales.

1.4. Cycle de vie d'un système d'information et historique

Si l'on observe le cycle de vie d'un système d'information, on constate différentes étapes de changements. Avant l'ère de l'informatique, toutes les organisations disposaient de fichiers conservés sous forme de papier et procédaient à des traitements manuels sur ceux-ci. Cela constituait leur système d'information.
L'apparition de l'informatique a conduit les organisations à transférer leurs données sur des fichiers informatiques afin d'en faciliter la gestion et manipulation. Ainsi, à chaque traitement manuel fut consacré une application informatique. Celle-ci nécessite des fichiers spécifiques ainsi que des ressources informatiques et humaines. Le nombre d'applications étant élevé (gestion des clients, gestion du personnel, gestion des stocks,...) le nombre de fichiers l'était également.
Ensuite est apparu le concept de base de données. On a dès lors remplacé individuellement les applications de traitements de données existantes en applications utilisant un système de gestion de base de données disposant de ses propres ressources. Mais vu la **diversité des applications**, un même système de gestion de base de données ne pouvait supporter les fonctionnalités de chacune d'elles. En effet, certains modèles de données conviennent mieux pour résoudre une catégorie de problème plutôt qu'une autre. Une prolifération de systèmes de base de données hétérogènes s'en suivit au sein d'une même organisation.
L'hétérogénéité des bases de données est également du à la **complexité des organisations**. En effet, dans une structure de grande envergure, chaque département tend à définir son propre système indépendamment des autres. Cela conduit implacablement à une hétérogénéité dans les systèmes de bases de données.
De plus la **fusion et l'acquisition** d'organisations sont également un facteur de cause à ce type de problème.
Cet historique met donc en évidence l'inévitable apparition du problème dans le cycle de vie de tout système d'information.
A présent que les causes du problème ont pu être identifiées, déterminons les motivations qui nous poussent à faire face à cette situation.

1.5. Motivations

Les motivations sont peu nombreuses mais d'une importance considérable. En effet, les **besoins croissants d'informations** se font sans cesse ressentir dans toute organisation. L'information est, dit-on, le pouvoir de décision d'une organisation. En effet, les tendances de mondialisation des organisations et de leurs marchés rendent leurs besoins informationnels de plus en plus importants pour s'assurer une position concurrentielle avantageuse. Il semble donc indispensable d'offrir à toute organisation les possibilités d'obtenir le plus d'informations pertinentes à leurs prises de décision. De plus **l'avènement des réseaux de télécommunications** n'a fait

qu'accroître ce souhait d'obtenir une meilleure information. Ces techniques offrent les moyens de communications entre matériels informatiques. Par conséquent, il est indispensable de permettre la communication entre systèmes de bases de données. Cette dernière permettra de répondre aux besoins d'interconnexion intra- et inter-organisationnelle. Ces deux facteurs constituent la motivation à la recherche d'une solution.
L'hétérogénéité des SBD ne permet en effet qu'un partage limité des informations car chacun de ces systèmes nécessite des connaissances spécifiques pour accéder à leurs données. Il semble illusoire d'imposer à un utilisateur d'apprendre une multitude de langages de manipulation et de modèles de données. De plus la multiplicité des bases de données induit des coûts de ressources et de maintenance considérables. Le nombre de bases de données croissant considérablement dans toute organisation, engendre des coûts supplémentaires. Il apparaît donc évident qu'une solution doit être trouvée pour faciliter le partage des informations au sein d'une même organisation, voire entre plusieurs organisations. La complexité du problème réside essentiellement dans la limitation d'un SGBD à ne reconnaître qu'un modèle et un langage.

1.6. Philosophies de résolution

Trois philosophies de solution se dégagent de cette analyse de la situation. On peut les classer sur un axe de complexité.

A. Philosophie du moindre effort

La solution consiste à imposer à l'utilisateur une interrogation individuelle des bases de données. A partir des données recueillies, l'utilisateur se charge de les trier pour éliminer la redondance. C'est donc la solution la plus simple mais également la plus irréaliste puisqu'elle impose à l'utilisateur une connaissance de tous les modèles et langages de manipulation des bases de données concernées. Face aux objectifs fixés dans la définition de la situation, cette philosophie ne laisse paraître aucune transparence de distribution et d'hétérogénéité.

B. Philosophie de la refonte complète: intégration physique des bases de données

Cette solution est la plus complexe dans la mesure où elle impose une re-conception complète du système d'information. Cela implique une re-définition des schémas, des bases de données, des applications et un ré-encodage complet des données. Vu la complexité des systèmes d'information existants et leur nécessité indiscutable, il est tout à fait utopique d'envisager une refonte complète du système d'information. Cette solution est impossible pour les raisons suivantes: importance des coûts de redéfiniton, la demande de requête globale n'est pas nécessairement permanente, le nombre de bases de données croît sans cesse et il arrive que des bases de données soient issues de l'extérieur d'une organisation, rendant la solution physiquement impossible.
De plus cette philosophie ne respecte pas l'autonomie des systèmes locaux.

C. Philosophie du juste milieu: intégration logique ou virtuelle des bases de données

Entre ces deux philosophies, on trouve des solutions plus raisonnables consistant à fournir à l'utilisateur un moyen d'accès uniforme et transparent à cette collection de bases de données. On parle d'intégration logique ou virtuelle dans la mesure où la base de données ne sera en fait constituée que d'un schéma et de relations avec les bases de données locales. Le chapitre 2 sera consacré à l'analyse des différentes architectures répondant sa cette philosophie.

Il semble opportun d'examiner ici la question des **standards** en matière de bases de données. En effet, on pourrait envisager d'imposer des standards à la définition de systèmes de bases de données, tant au niveau des modèles et des SGBD que des langages. Cela faciliterait considérablement le problème puisqu'il disparaîtrait. Mais la définition de standards est malheureusement une tâche bien plus complexe qu'on ne peut l'imaginer. La spécification de standards est tout simplement impossible pour trois raisons. La première concerne la **spécificité des applications**. On ne peut en effet imposer l'utilisation d'un SGBD particulier. Celui-ci serait incapable de résoudre toutes les catégories de problèmes. On ne résout pas un problème d'informatisation de gestion de personnel comme on résout un problème d'informatisation d'un processus de conception de véhicules automobiles. Ces deux problèmes nécessitent des outils différents. La seconde raison concerne la **liberté de mise sur le marché de produits différents**. Les standards limitent la diversification des produits mais également l'innovation technologique. la troisième raison est que la définition de standards nécessite une **technologie mûre et qui a fait ses preuves**. Le domaine des bases de données est en constante évolution de nos jours. Ces raisons montrent bien l'impossibilité actuelle de définir des standards. Les systèmes relationnels ont fait largement l'unanimité pendant de nombreuses années et aujourd'hui la tendance serait plutôt du côté des modèles orienté-objets. Alors, à quand un standard?

CHAPITRE 2: ARCHITECTURES DE SBDH: ETAT DE L'ART

2.1. Introduction

Ce second chapitre sera consacré à la présentation de différentes architectures de systèmes de bases de données hétérogènes. Nombreux sont les problèmes sous-jacents à la résolution intégrale du problème des BDH. En effet, dans ce contexte il est nécessaire de se poser une multitude de questions pour essayer d'orienter la recherche d'une solution. Dans un premier temps, sans entrer dans le détail, on peut s'arrêter les questions suivantes:

Comment offrir à l'utilisateur une vue intégrée logique de la collection de bases de données?

Face à cette vue intégrée, l'utilisateur va introduire des requêtes. Comment va-t-il introduire ces requêtes et comment répondre à ces requêtes étant donné qu'il n'existe pas de réelle base de données intégrée?

Comment assurer les correspondances entre le SGBDH et les SBD locaux?

Comment adapter la gestion des transactions au niveau du SGBDH pour que l'intégrité des BD locales soit respectée?

Comment régler la concurrence entre les requêtes locales et les requêtes globales, entre les différents utilisateurs?

Comment assurer une certaine autonomie aux SBD locaux?

Comment résoudre les incompatibilités de données?

Comment faire face à l'hétérogénéité des schémas, des modèles, des langages entre les différents SBD locaux?

En somme comment assurer à l'utilisateur la **triple transparence** de distribution, d'hétérogénéité et d'autonomie?
Il existe différentes architectures pour essayer de répondre à ces exigences. Nous verrons que ces architectures se différencient par leur **degré de prise en compte de ces exigences.**
A partir des caractéristiques de distribution, d'hétérogénéité et d'autonomie qui ont été définies auparavant, [OZSU 91] a établi une typologie (figure 2.1) des systèmes constitués de plusieurs SBD.

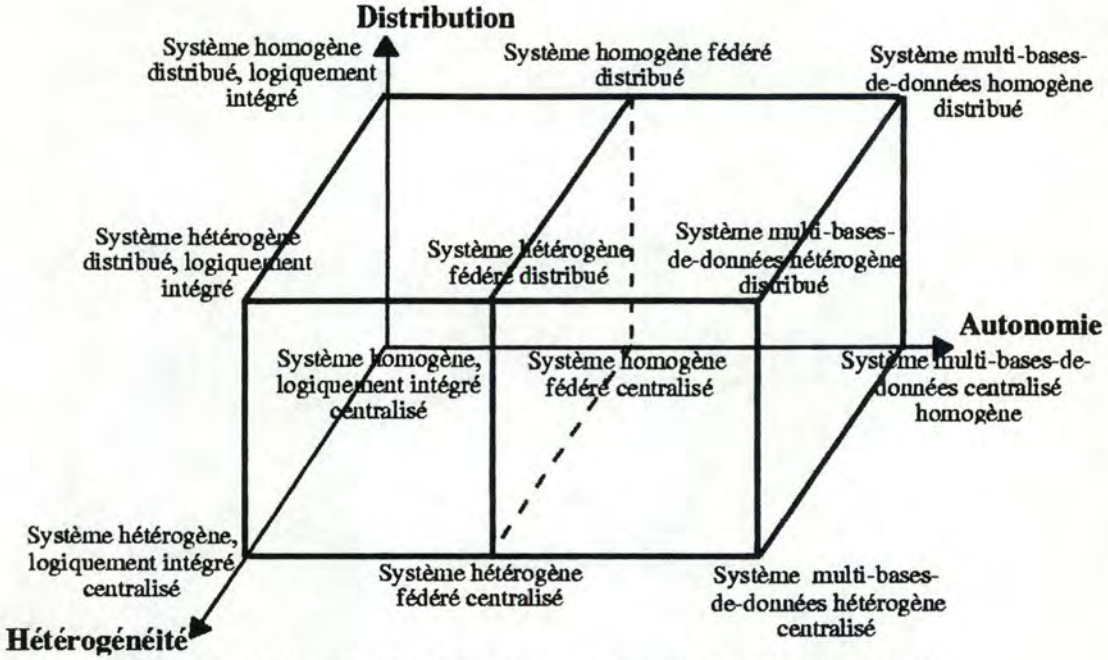


Figure 2.1: Typologie tri-dimensionnelle selon [OZSU 91]

La représentation graphique de cette typologie parle d'elle-même. Notons simplement que la distinction existant entre les systèmes logiquement intégrés, fédérés et multi-bases-de-données se situe au niveau de l'autonomie et de la méthode d'intégration des schémas locaux. La corrélation existant entre le degré d'autonomie et celui d'intégration apparaîtra au fil de l'analyse des architectures.
Dans le cadre de notre étude des BDH, nous nous situons dans le plan des systèmes hétérogènes. Nous pouvons restreindre notre analyse à la droite des systèmes hétérogènes et distribués. En effet ces deux caractéristiques vont de pair dans le contexte des bases de données hétérogènes. De plus, le souhait de développer des SGBD distribués est en pleine expansion. Nous aurons donc trois approches différentes à analyser: l'approche intégrée, multi-bases-de-données et fédérée.

Des classifications orientées sur une des trois dimensions ont également été proposées et notamment celle de [SHETH 90] qui ne tient compte que de la dimension d'autonomie. Cette classification nous permettra une analyse plus approfondie de l'approche fédérée puisqu'elle distingue encore trois catégories de systèmes fédérés: les systèmes faiblement couplés, les systèmes fortement couplés mono-schéma et les systèmes fortement couplés multi-schémas.

Examinons à présent les critères d'analyse des architectures.

2.2. Critères d'analyse

L'étude des différentes approches de résolution sera orientée selon **deux axes**. Premièrement, on exposera l'**architecture des schémas** de façon générale et ensuite nous présenterons des études de cas se rapportant à cette approche.

A. Axe "architecture des schémas"

L'analyse de l'architecture des schémas va permettre d'identifier les différents **niveaux** de schémas utilisés dans le SBDH. L'analyse selon cet axe nous permettra de percevoir les différents niveaux d'**abstraction** de l'architecture ainsi que les **liens** existants entre ceux-ci. Cette architecture nous servira de **référence** pour les études de cas.
L'étude des propriétés et des méthodes de construction de certains schémas nous permettra d'examiner les **propriétés générales** de l'architecture face aux objectifs que nous nous sommes fixés.
Afin d'examiner l'architecture des schémas, on se basera sur celle de l'**ANSI/SPARC**, bien connue de tous. Cette dernière distingue trois niveaux d'abstraction: **interne**, **conceptuel** et **externe**.

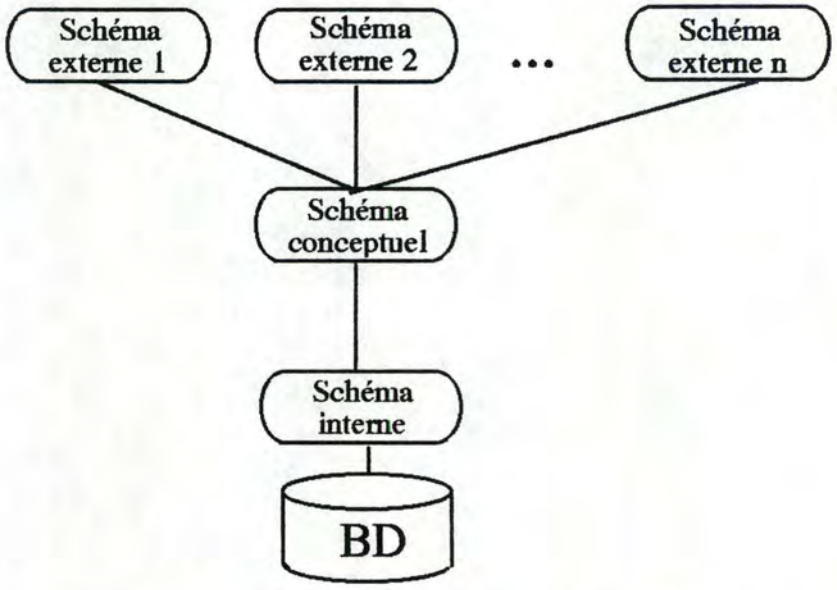


Figure 2.2 : Architecture de l'ANSI/SPARC

Le **schéma conceptuel** consiste en un ensemble d'objets qui fournissent une description logique (structures des données et liens entre ces structures) de la base de données, c'est-à-dire indépendante des paramètres techniques de la BD. Cette perception permet d'acquérir une certaine stabilité par rapport au système réel.
Le **schéma interne** est le schéma décrivant la base de données de manière physique c'est-à-dire les caractéristiques d'implantation de la base de données telles que la position des enregistrements, les index,... Ce schéma vérifie les contraintes du SGBD. Ce niveau constitue donc la vue de l'informaticien.
Le **schéma externe** constitue la vue de l'utilisateur. Elle est définie comme un sous-ensemble du schéma conceptuel. On accepte de multiples schémas externes du fait de la variété des besoins des utilisateurs.
Il existe des correspondances entre le niveau conceptuel et interne ainsi qu'entre le niveau externe et conceptuel; en jargon d'informaticien, ces correspondances sont appelées **mapping**.

B. Axe "études de cas"

Pour chacune des approches, nous exposerons l'**architecture complète** d'un ou plusieurs systèmes réels ou prototypes pour la comparer avec l'architecture de référence de l'approche. Ces études de cas nous amènerons à étudier plus particulièrement la démarche de résolution du problème des BDH mais également les entorses vis-à-vis de l'architecture de référence des schémas. Pour faciliter la lecture des schémas, nous utiliserons la terminologie de l'architecture de référence mais nous ne manquerons pas de donner les correspondances (sous forme de tableau) avec celle proposée par les auteurs de ces architectures.

Analysons à présent les différentes architectures.

2.3. L'approche intégrée

L'idée centrale de cette solution est basée principalement sur l'architecture des systèmes de bases de données distribuées. Elle consiste en effet en la définition d'un **schéma conceptuel global unique** intégrant l'ensemble des schémas **conceptuels** des bases de données locales.
Les opérations de manipulations des BD locales sont exprimées dans un langage universel et ensuite médiatisée par le schéma global. Ce dernier offre l'illusion d'avoir une base de données centralisée unique. Les utilisateurs du schéma global ne sont pas conscients des conflits existants entre les SBD locaux car la résolution explicite de ceux-ci sont spécifiées à l'avance.
Deux **remarques fondamentales** apparaissent: l'ensemble des applications définies sur les schémas locaux ne sont plus opérationnelles et le schéma global est difficile à définir étant donné l'hétérogénéité des SBD locaux. De plus toute modification d'un schéma local ou l'adjonction d'un nouveau SBD local contraint à reconstruire le schéma global.

A. Architecture des schémas

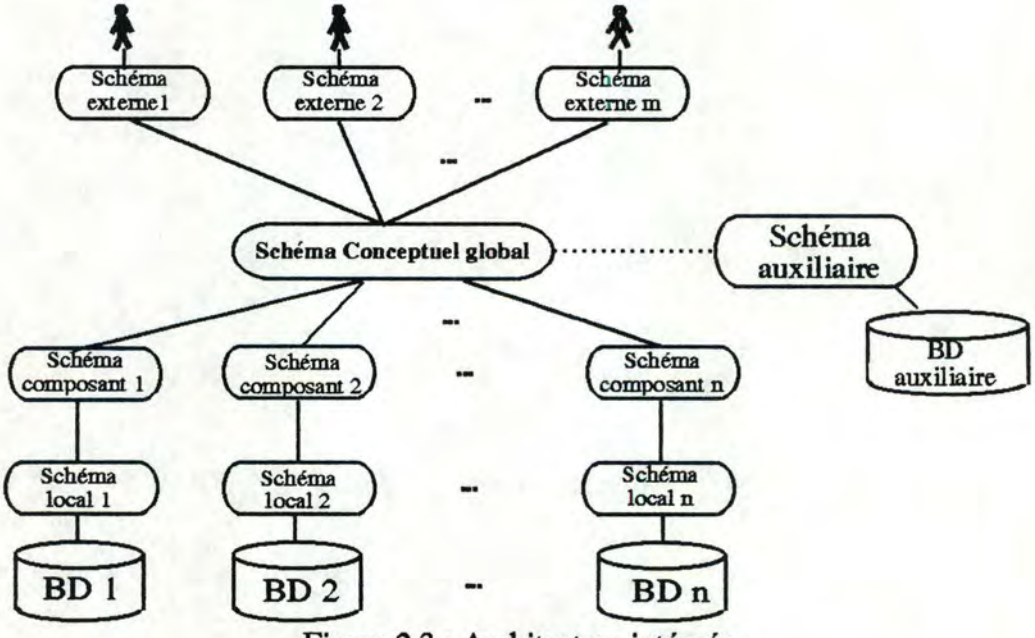


Figure 2.3 : Architecture intégrée

Cette architecture compte donc quatre niveaux : le local, le composant, le global et l'externe.

Le niveau **local** reprend le schéma **conceptuel** de la base de données locale; celui-ci est donc conforme au modèle conceptuel de la BD locale. Ainsi les schémas locaux 1 à n peuvent être exprimés dans des **modèles différents.**

Le niveau **composant** reprend le schéma **composant** c'est-à-dire la vue du schéma local dans un modèle de données commun, encore appelé **modèle canonique.** A ce niveau, les n schémas composants sont conformes à un et **un seul modèle de données.** A chaque schéma de ce niveau correspond donc un schéma du niveau local. On a donc une **correspondance biunivoque** entre schéma composant et schéma local.

Le niveau **conceptuel global** constitue une **vue complètement intégrée** des schémas composants. Ceci signifie donc qu'à chaque élément de chacun des schémas composants correspond un[1] élément du schéma conceptuel global. Le modèle de données utilisé est naturellement le modèle canonique pour des raisons évidentes.

Le niveau **externe** permet aux utilisateurs de définir des vues sur le schéma conceptuel global comme il est habituel de le faire dans les SBD traditionnels.

Certaines architectures répondant à cette approche ont un **type de schéma supplémentaire.** Ce type de schéma, destiné à stocker les correspondances entre le schéma conceptuel global et les schémas composants, est généralement nommé **schéma auxiliaire** [DAYAL 84]. Son utilisation sera discutée ultérieurement. D'autres types de systèmes intègrent ces correspondances au sein du schéma global [MOTRO 87].

Il faut remarquer que généralement, on utilise le même modèle de données pour les niveaux global et composant. Mais il n'est pas inhabituel de trouver des schémas externes dans des modèles de données différents du modèle canonique.

Cette architecture ne permet plus que l'accès par l'intermédiaire du schéma global ou des vues externes définies sur ce dernier. Il n'est désormais plus possible d'interroger les SBD locaux après la transformation d'architecture, le contrôle étant laissé au SGBDH via son schéma global.

Cette approche offre les avantages de bénéficier d'une vue uniforme de l'ensemble des bases de données et donc de permettre à la fois les opérations de consultation et de modification tout en respectant l'intégrité des bases de données locales. Mais ces avantages se font au détriment de l'autonomie des SBD locaux. Une fois le processus enclenché, les SBD locaux perdent leur autonomie et le seul accès se fait désormais par le schéma global.

L'architecture du système CARNOT proposé dans [COLLET 91] est basée sur cette approche d'intégration globale, mais présente des caractéristiques particulières.

1. Cette correspondance peut être du type 1-1, 1-N, N-1, N-N

B. Etude de cas: CARNOT

Ce système a été conçu pour permettre la consultation et la mise à jour de multiples bases de données partageant des incompatibilités relatives à la sémantique, à la syntaxe, au matériel, au logiciel d'exploitation, aux structures de données logiques et physiques. Les objectifs de ce projet étaient donc de développer une méthode d'intégration de systèmes développés séparément et présentant des incompatibilités, de permettre l'accès à ces systèmes pour consultation et modification tout en respectant l'intégrité des BD locales.

L'architecture de ce système est exposée à la figure 2.4.

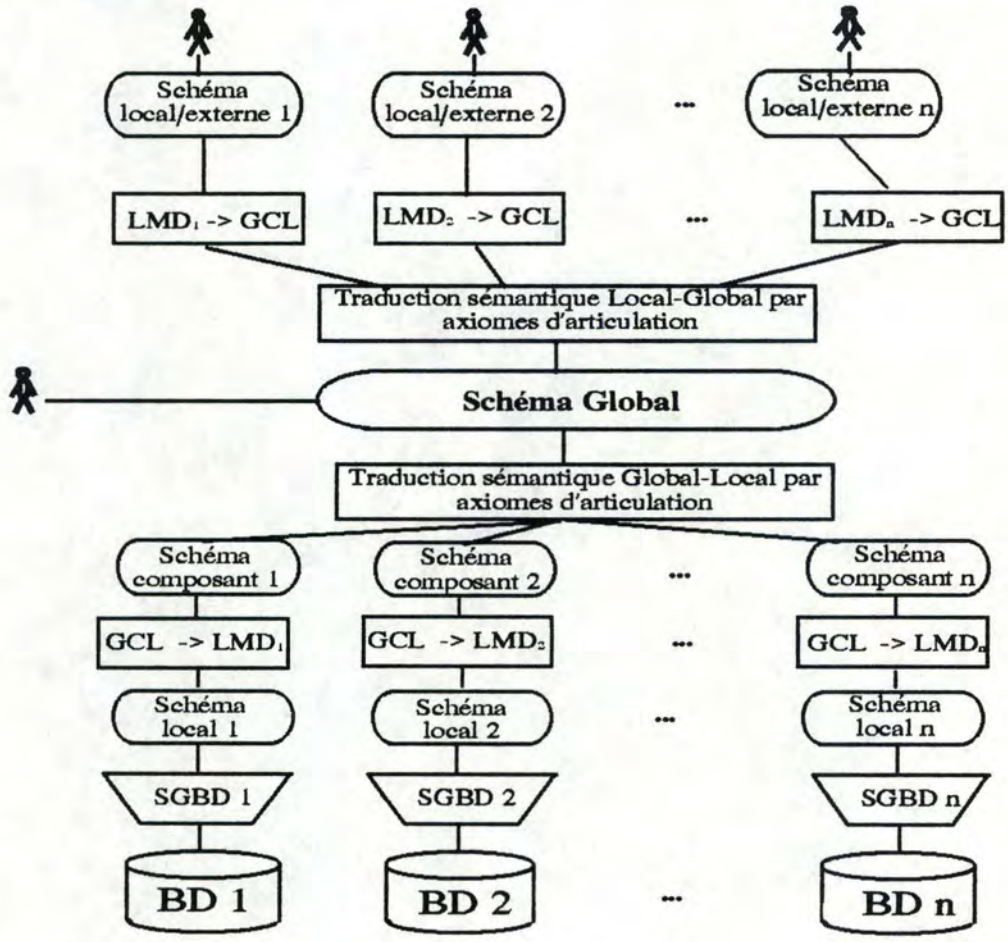


Figure 2.4: Architecture de CARNOT

Avant d'expliquer le fonctionnement de CARNOT, exposons la correspondance entre la terminologie de CARNOT et celle de l'architecture de référence, les modèles et langages de données utilisés par CARNOT (Tableau 2.1).

Tableau 2.1			
Référence	CARNOT	Modèle	LMD
Sch. Local	Sch. Local	Modèle M_i	LMD_i
Sch. Composant	Sch. Local GCL	Modèle BD CYC	GCL
Sch. Global	Vue Globale	Modèle BD CYC	GCL
Sch. local/externe	Vue Locale	Modèle M_i	LMD_i

Observations

1. La première observation concerne l'équivalence entre le premier et le dernier niveau de schéma de l'architecture. Il y a une parfaite relation d'égalité entre ces deux types de schémas. Cette relation permet aux applications locales existantes de rester totalement opérationnelles, mis à part qu'elles seront appliquées sur l'ensemble des schémas locaux. Cette technique est en réalité un subterfuge pour ne pas dire que l'accès local est supprimé. Ainsi deux techniques d'accès sont disponibles: directement par le schéma global en utilisant le langage GCL ou par les schémas locaux/externes en utilisant les langages des SBD locaux correspondants. De ce fait, l'utilisateur est contraint à l'apprentissage d'un nouveau langage de manipulation GCL et du modèle de Cyc s'il souhaite étendre ses opérations de manipulation.

2. La seconde observation concerne le schéma global. Celui-ci est en réalité un schéma déjà existant à partir duquel on tente d'établir des correspondances avec les schémas locaux. Le schéma global est en effet celui de la base de connaissances CYC, utilisant le modèle de structuration de connaissances, basé sur les concepts de FRAME et de SLOT. Un FRAME est un élément d'information associé à un concept particulier du réel. Il mentionne les propriétés de ce concepts par des SLOTS et reprend également des procédures applicables aux FRAMES. Il existe des mécanismes d'héritage de propriétés et de procédures entre FRAME.
Ce modèle admet comme seul langage de manipulation, le langage GCL (Global Context Language) qui est basé sur la logique du premier ordre.

3. L'intégration des SBD locaux est réalisée par des mappings distincts entre chaque schéma local et le schéma global. Chaque mapping consiste en une traduction syntaxique et une traduction sémantique. La **traduction syntaxique** consiste en une transposition des schémas locaux dans le formalisme du schéma global, en l'occurrence celui de CYC mais également en une traduction dans le langage global (GCL) des requêtes exprimées en langage local (LMD_i -> GCL) et vice-versa (GCL ->LMD_i). La **traduction sémantique** se charge de mettre en correspondance les éléments de chaque schéma local avec ceux du schéma global, ces deux types de schéma étant représentés selon un formalisme unique. Les mappings sémantiques entre schémas composants et global sont décrits au moyen d'axiomes d'articulation.

Un **axiome d'articulation** est défini comme une assertion d'équivalence entre deux éléments et permet à la fois un mapping sémantique entre schémas et un mécanisme de traduction entre

schéma global et schémas locaux. Ces axiomes d'articulation ont la forme suivante :

$ist(G, \alpha) \Leftrightarrow ist(L_i, \beta)$ avec G schéma global
L_i schéma local i
α et β des expressions logiques portant respectivement sur des éléments des schémas G et L_i
"ist" prédicat "**is true in the context**" et se lit comme suit :"l'expression logique α du schéma G est équivalent à l'expression logiqueB du schéma L_i"

Ainsi pour chaque schéma local, on aura à définir un ensemble d'axiomes d'articulation.

Le processus d'intégration et la définition des axiomes d'articulation se font en **trois étapes** : représentation automatique des schémas locaux dans le formalisme Cyc, matching des concepts des schémas locaux dans le schéma global de la BD Cyc et construction des axiomes d'articulation. La **première phase** consiste en réalité en la traduction des schémas locaux dans un formalisme unique, qui est celui de Cyc. Cette phase consiste donc à définir des schémas composants dont le modèle canonique est celui de Cyc. La base de cette traduction est une table de correspondance entre les concepts des modèles des systèmes locaux et ceux de Cyc. La **seconde phase** consiste à trouver dans le schéma global de la BD Cyc l'élément correspondant à celui du schéma composant. Cette phase se déroulera en interaction avec l'administrateur de la base de données qui pourra le cas échéant opter pour l'une ou l'autre alternative en cas de conflit entre concepts. En cas d'échec dans la recherche, l'élément sera créé. La **dernière phase** consiste en la définition des axiomes.

4. Le schéma global est implémenté comme la combinaison de trois processeurs:

- un **générateur de transactions**, qui assure la décomposition des requêtes globales en requêtes locales

- un **distributeur de transactions**, qui s'occupe de distribuer les requêtes locales dans le formalisme local aux SBD correspondants

- un **assembleur de résultats**, qui sur base des résultats des requêtes locales compose la réponse à la requête globale

5. Plutôt que de devoir refaire le schéma global à chaque arrivée d'une nouvelle BD à intégrer ou d'une modification d'une BD déjà intégrée, cette solution utilise un **schéma global existant**, la base de données Cyc. Ainsi les schémas des SBD locaux sont comparés de façon indépendante et fusionnés avec la BD Cyc. En effet cette BD présente les avantages de disposer de 50000 entités et relations exprimées en termes de "frame", de couvrir de nombreux sujets du monde réel, d'être riche en mécanismes d'abstraction, de représentation des données et d'inférence. Toutes ces caractéristiques rendent plus aisée la phase d'intégration et de maintenance du schéma

global.

6. Le processus d'intégration de schémas n'utilise pas seulement une description structurelle des schémas locaux pour résoudre les différences sémantiques mais également toutes informations sur les schémas (structures des données, contraintes d'intégrité, opérations autorisées), les systèmes locaux (services fournis, modèle utilisé, langage utilisé) et l'organisation des systèmes locaux (règles régissant l'utilisation des systèmes locaux).
L'utilisation du schéma global Cyc rend la tâche de l'utilisateur assez complexe vu la taille du schéma global et le nombre d'éléments inutiles pour l'utilisateur. De plus cela nécessite l'apprentissage du langage GCL. C'est pourquoi l'architecture reprend un niveau de schémas locaux/externes identiques aux schémas des SBD locaux.

7. La limitation des schémas locaux/externes ne permet pas d'en définir d'autres sur le schéma global mais permet de conserver les applications existantes excepté que leur domaine d'application est devenu l'ensemble des bases de données locales. Cette limite est très gênante dans la mesure où le schéma global est de grande taille.
L'autonomie des SBD locaux n'est donc pas respectée car le partage contrôlé des données n'existe pas et la possibilité d'accéder localement n'est pas autorisée.

Analysons à présent comment cette architecture traite une requête posée sur un schéma externe quelconque.
Soit une requête R_0 exprimée dans le formalisme du langage L associé au modèle M du schéma externe S. Cette requête est traduite simplement dans le langage GCL en une requête R_1. Ensuite les arguments de R_1 sont remplacés (à l'aide des axiomes d'articulations) par leurs correspondants du schéma global, fournissant ainsi la requête R_2. Le générateur de transaction décompose (en utilisant les axiomes d'articulations) cette requête R_2 en un ensemble de sous-requêtes $R_{21}, R_{22},.., R_{2P}$ destinées aux SBD locaux. Le distributeur de transactions attribue les sous-requêtes aux processeurs de traduction des SBD respectifs. Ces processeurs traduisent individuellement les sous-requêtes pour obtenir $R_{31}, R_{32},...,R_{3P}$, requêtes conformes aux langages respectifs de chaque SGBD local. Ces requêtes sont alors transmises aux SGBD locaux pour exécution individuelle. Ces derniers renvoient les résultats $R_{41}, R_{42},..., R_{4P}$ indépendamment les uns des autres pour être traduits dans le format de Cyc, $R_{51}, R_{52},..., R_{5P}$. Ces derniers sont transmis à l'assembleur de résultats pour assemblage en un résultat R_6 qui est converti ensuite dans le format du modèle M et conformément au schéma S initial.

2.4. L'approche multi-bases-de-données

A l'opposé de l'approche intégrée où l'on construit un schéma global intégré, l'idée de cette approche est basée sur l'**absence totale de définition d'un schéma global**. Cette solution consiste donc en la définition d'un **langage de manipulation multi-bases** ou en d'autres mots un langage permettant de manipuler plusieurs bases de données simultanément.
Les requêtes porteront sur plusieurs bases de données **simultanément** et **explicitement**. Cela signifie que l'utilisateur spécifiera dans ses requêtes les bases de données auxquelles il désire accéder. Un processeur se chargera de décomposer la requête et de composer la réponse globale

au départ des réponses locales multiples.
Cette approche n'a pas emporté de grand succès vu qu'elle ne répond que très peu aux exigences imposées en matière de base de données hétérogènes et notamment à la transparence de localisation et de distribution. C'est pourquoi nous nous contenterons de donner un aperçu de l'architecture des schémas et un exemple de mise en oeuvre possible de cette approche.

A. Architecture de schémas

L'architecture se compose de **deux** niveaux: le niveau **local** et le niveau **composant.** Le niveau local reprend le schéma conceptuel de la base de donnée locale et le niveau composant la traduction de ce schéma dans un formalisme commun.

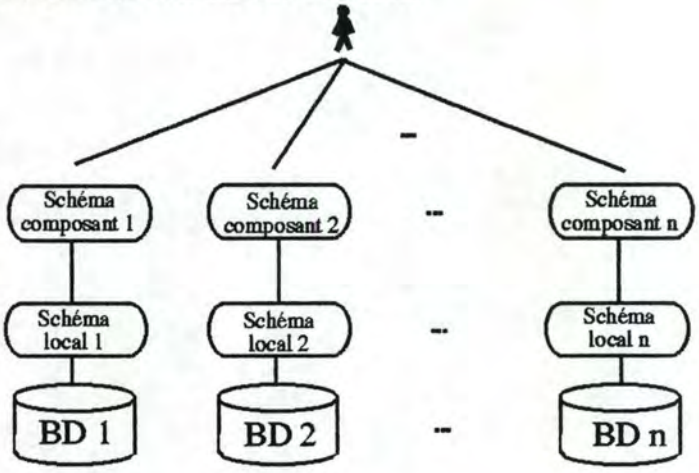


Figure 2.5: Architecture multi-bases-de-données

B. Etude de cas

L'architecture de ce système est présentée à la figure 2.5. et offre un langage de manipulation telle que l'utilisateur puisse manipuler plusieurs bases de données simultanément. La résolution de ces requêtes multi-bases consiste en la création d'une vue intégrant les schémas désignés. Cette vue est temporaire puisqu'une fois la requête terminée, elle sera détruite. Ce langage fournit une réponse intégrée à la requête globale.
En prenant le modèle relationnel comme modèle canonique, on obtiendrait des requêtes de type SQL telles que:

```
select *
from A@BD1, B@BD2
where A.num = B. numéro
```

où A. nom signifie l'attribut "nom" du schéma A, et A@BD1 désigne la table A de la base de données BD1.

Cette approche impose donc à l'utilisateur de connaître la distribution et la localisation des

données, mais offre l'avantage d'assurer aux BD locales une autonomie complète.

2.5. L'approche fédérée

Cette approche [SHETH 90] est celle qui a été la plus largement suivie pour l'élaboration de prototypes de SBDH.
Elle consiste à définir un système de base de données fédérées (SBDF) c'est-à-dire un système constitué d'une collection de SBD autonomes qui participent à une fédération pour permettre le partage partiel et contrôlé de leurs données. Ainsi chaque SBD local garde le contrôle sur les données qu'il gère.
Cette approche représente donc un compromis entre les deux approches précédentes. Ainsi, aucune des applications existantes sur les SBD locaux ne seront affectées par cette nouvelle architecture. La préservation de cette autonomie des SBD locaux est réalisée grâce à la distinction entre *accès local* et *accès global* offerte par le SBD fédéré.

L'idée de base de cette solution consiste à définir un(des) **schéma(s) fédéré(s)**. Ce type de schéma constitue la **vue intégrée** d'un **sous-ensemble utile** de **schémas exportés**. L'architecture va nous permettre de comprendre cette définition.

La typologie de [SHETH 90] distingue trois types de SBD Fédéré selon la méthode d'intégration adoptée et les responsables de la gestion du schéma fédéré: le SBDF faiblement couplé, le SBDF fortement couplé mono-schéma, le SBDF fortement couplé multi-schémas.

Dans le **système faiblement couplé**, c'est l'utilisateur qui est responsable de l'intégration des schémas et de la gestion du schéma fédéré. Il n'y a aucun contrôle effectué par le système fédéré ou l'administrateur. Cette architecture accepte plusieurs schémas fédérés.

Dans le **système fortement couplé mono-schéma**, ce sont la fédération et l'administrateur du schéma fédéré qui sont responsables tant au point de vue de l'accès aux SBD locaux que de la maintenance. L'intégration est réalisée de manière automatique ou semi-automatique. Ce système n'autorise la création que d'un seul schéma fédéré reprenant l'ensemble des schémas exportés.

Le **système fortement couplé multi-schémas** reprend les mêmes caractéristiques que le système précédent mais accepte une multiplicité des schémas fédérés et donc n'impose rien sur le contenu des schémas fédérés.

A. Architecture des schémas

Les nombreuses solutions apportées au problème des BDH ont permis d'établir une architecture de schémas à **cinq niveaux** d'abstraction. Nous verrons par la suite que cette architecture est générale et que certaines solutions se limitent à 3 ou 4 niveaux et que d'autres insèrent des

schémas supplémentaires. Ces limitations et extensions seront notamment exposées lors des études de cas.
L'exposé d'une architecture de référence (figure 2.6) nous permettra de mieux comparer les SBD hétérogènes qui ont été développés.

Le **schéma local** est le schéma conceptuel d'un SBD local. Ce schéma est exprimé dans le modèle conceptuel du SBD local. Les schémas locaux des SBD locaux peuvent donc être exprimés dans des modèles différents.

Le **schéma composant** est dérivé par traduction du schéma local dans un modèle de données commun et appelé modèle canonique. A ce niveau, on dispose donc de n schémas dont le modèle de données est identique.

Le **schéma exporté** est une partie du schéma composant destiné à être partagé avec les utilisateurs non locaux. Ce type de schéma facilite le contrôle et la gestion de l'*autonomie d'association.*

Le **schéma fédéré** constitue une vue intégrée de multiple schémas exportés. Ce schéma contient de l'information sur la distribution des données si celle-ci n'est pas reprise dans un schéma séparé, généralement appelé schéma de distribution. Le schéma fédéré a évidemment comme fonction d'assurer une transparence de distribution/localisation des données.

Le **schéma externe** est un schéma défini par l'utilisateur ou l'application. Ce niveau a les fonctionnalités identiques à celui du schéma externe de l'architecture de l'ANSI/SPARC, notamment le contrôle d'accès, la personnalisation de la base de données et l'ajout éventuel de contraintes supplémentaires.

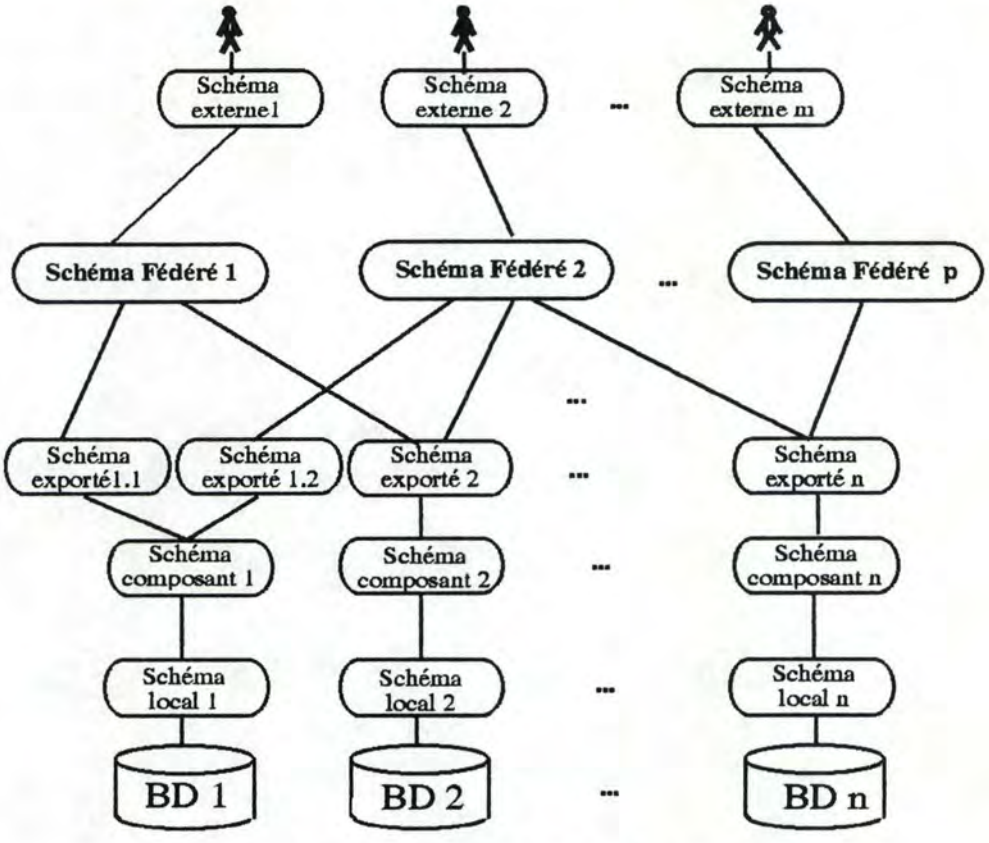


Figure 2.6 : Architecture fédérée de référence

Les trois types d'architecture fédérée se distinguent au niveau du schéma fédéré, par la manière de le construire et de le gérer. Ces deux caractéristiques influent sur les capacités offertes par l'architecture.

B. L'architecture fédérée faiblement couplée

Cette architecture est typiquement celle de la figure 2.6. Elle permet la définition de plusieurs schémas fédérés à partir des schémas exportés, seules parties de schémas visibles et disponibles par les utilisateurs.
La **définition des schémas exportés** se fait indépendamment des souhaits des utilisateurs. Seuls les administrateurs des bases de données locales ont pouvoir de définir les parties de schémas qu'ils partageront avec les utilisateurs de la fédération.
Le **processus d'intégration** suivi pour construire un schéma fédéré se compose de trois phases, toutes étant sous le contrôle de l'utilisateur:

1. Consultation des schémas exportés disponibles pour déterminer les schémas utiles pour l'utilisateurs.
2. Définition du schéma fédéré en utilisant un interface d'intégration, tel un langage d'intégration et de restructuration comme ceux définis par [MOTRO 87] ou [DAYAL 84]. L'utilisateur doit comprendre la sémantique des schémas et résoudre les conflits entre les divers schémas

à intégrer. Pour se rendre la tâche plus facile, l'utilisateur peut disposer des dictionnaires des SBD locaux.
3. L'utilisateur donne un nom à son schéma qui est stocké sous le compte de l'utilisateur de la fédération.

Tout SBDF répondant à cette architecture est qualifié de **dynamique** pour deux raisons:

1. Le schéma fédéré est crée "sur le tas" par le futur utilisateur de ce schéma et répond donc à ses exigences. Ce type de processus permet de définir le schéma fédéré selon la sémantique que l'utilisateur perçoit au travers de ces schémas exportés. L'utilisateur définit précisément les relations entre les objets des schémas exportés.
2. Ce processus de définition de schémas fédérés n'est en rien déterministe et permet ainsi une multiplicité de sémantiques. On peut donc définir une multitude de schémas intégrés basés sur les mêmes schémas exportés et nul ne serait identique.

Cette dynamique de l'approche a comme avantage la définition personnalisée du schéma fédéré et facilite donc la définition du SGBDF qui doit remplir peu de fonctionnalités vu que l'utilisateur est fortement impliqué dans la définition de la solution. De plus, la technique des schémas exportés laisse aux systèmes locaux un contrôle important sur les données qu'ils gèrent. Par contre, l'inconvénient majeur résidant dans cette approche se situe au niveau de la tenue des mises-à-jour des bases de données. En effet, comment assurer une intégrité des bases de données locales quand on peut définir de multiples schémas fédérés dont les sémantiques peuvent diverger. Ce genre d'approche n'est pas du tout adéquat lorsque l'on souhaite que tous les utilisateurs voient l'ensemble des bases de données de manière uniforme.
On peut donc déduire de ces remarques que cette architecture s'applique bien si deux conditions sont vérifiées:

1. Bonnes connaissances de la part des utilisateurs des systèmes locaux et du processus d'intégration
2. Utilisation du SBDF en consultation uniquement, ce qui signifie qu'aucune mise-à-jour par l'intermédiaire des schémas fédérés ne sera acceptée au risque d'endommager considérablement l'intégrité des BD locales.

On peut donc qualifier cette architecture de "**Read-only, multi-schemata, user-oriented approach**" vu ses caractéristiques.

B.1. Etude de cas: MRDSM [LITWIN 86]

MRDSM est l'abréviation du système Multics Relational Data Store Multidatabase.
Il s'agit d'un système permettant d'intégrer des bases de données relationnelles conformes au modèle MRDS(Multics Relationnel Data Store) au SGBD associé. Il est néanmoins possible d'intégrer des BD ne répondant pas directement à ce critère mais présentant une vue conforme à

MRDS. Toute la problématique inhérente à la traduction de schémas, de requêtes et de données entre MRDS et le modèle quelconque n'est pas du ressort de cette architecture. Il s'avère donc que MRDSM est assez limité dans l'éventail des systèmes locaux qu'il admet. Au travers de cette architecture, nous remarquerons l'importance accordée à l'autonomie des systèmes locaux tant au niveau du partage des schémas qu'au niveau des accès locaux. En effet, la possibilité offerte par MRDSM de définir des schémas exportés, en réalité des vues MRDS, permet aux systèmes locaux de conserver leurs droits sur les données qu'ils gèrent ainsi que de limiter les opérations permises sur ces données.

La figure 2.7 illustre l'architecture globale de MRDSM en accord avec la terminologie adoptée par l'architecture de référence proposée à la figure 2.6. Le tableau 2.2. expose les correspondances de terminologie.

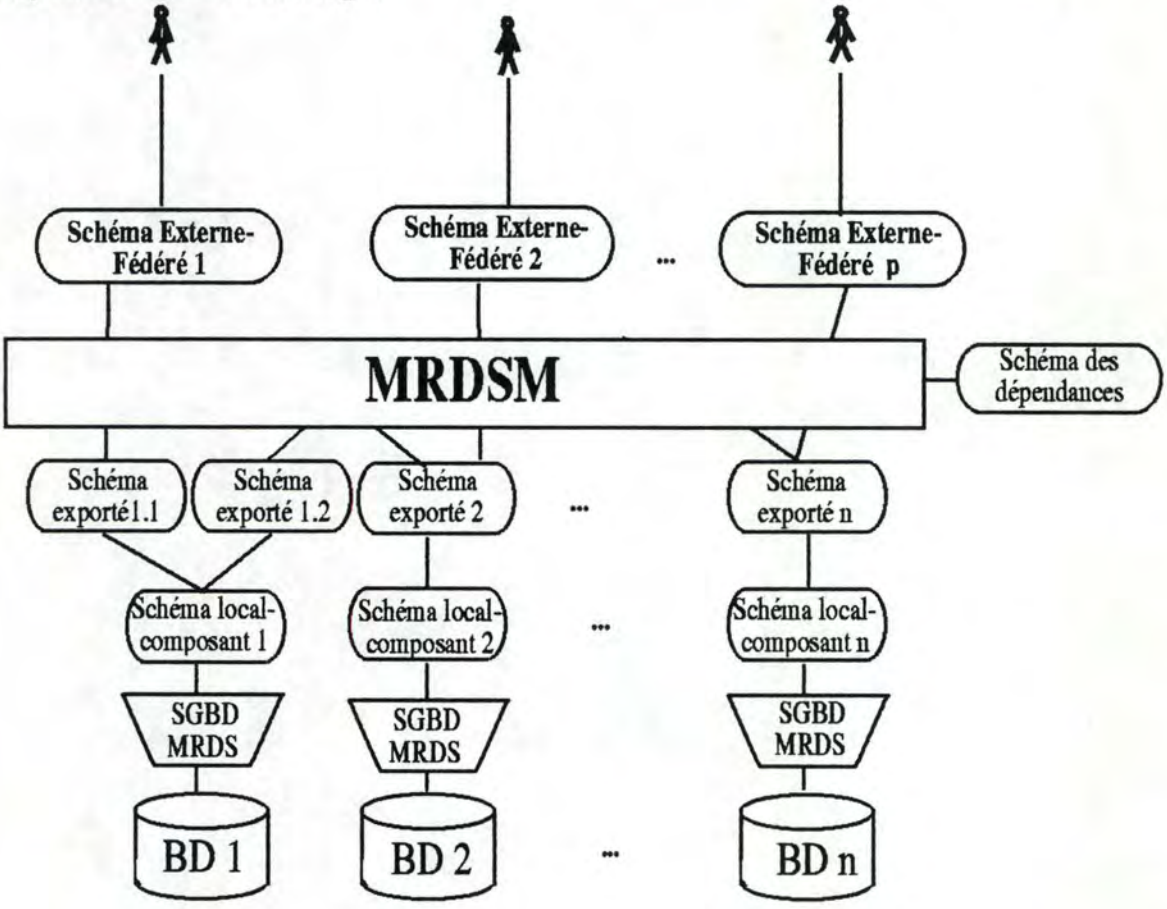


Figure 2.7. : Architecture de MRDSM

Tableau 2.2

Référence	MRDSM	Modèle	LMD
Sch. local/Comp	Vue relationnelle	relationnel MRDS	MDSL
Sch. Exporté	Sous-schéma	relationnel MRDS	MDSL
Sch. externe/fédéré	Schéma multi-bd	relationnel MRDS	LMD MRDSM

MRDSM ne gère pas la multiplicité des modèles au niveau local, c'est pourquoi les niveaux composant et local sont regroupés en un seul.
Le schéma externe-fédéré est crée par l'utilisateur à l'aide d'un langage spécifique. On ne distingue dès lors pas la différence entre schéma externe et fédéré puisqu'ils sont constitués à la convenance de l'utilisateur. Remarquons que seuls les schémas exportés peuvent être consultés par l'utilisateur pour définir son schéma fédéré.

La présence du **schéma des dépendances** est essentielle dans cette architecture. En effet, les administrateurs des bases de données locales définissent des dépendances entre les bases de données locales. Celles-ci, stockées dans le schéma des dépendances, permettent de lier les schémas coopérants. Trois types de dépendances sont proposées:

- les **dépendances de manipulation,** assurant l'intégrité inter-bases: exemple: l'insertion d'un "client" dans la BD 1 implique l'insertion de ce même "client" dans la BD 2. Ces dépendances sont définies en fonctions des opérations (insertion, suppression,...) effectuées sur certains éléments (clients, commandes,...) des schémas.

- les **dépendances de sécurité - confidentialité** des données: elles permettent de vérifier que la confidentialité des données n'est pas en péril lors des consultations et/ou modifications des données.

- les **dépendances d'équivalence:** permettent d'exprimer les relations entre identifiants dont l'égalité de valeurs conduit à la représentation d'un même objet. Ces dépendances permettent d'effectuer des jointures implicites c'est-à-dire non spécifiées dans la requête de l'utilisateur.

Les utilisateurs ont à leur disposition un **langage assertionnel** , MDSL (Multics Data Sub-Langage), au moyen duquel ils expriment des opérations de consultation et de mise-à-jour des données. Ce langage est basé sur le calcul de tuple et offre entre autre la possibilité de donner un nom logique à une collection de bases de données, de réaliser des jointures inter-bases, de réaliser des échanges de données entre les bases de données. La simplicité des expressions de ce langage est destinée à faciliter la tâche de l'utilisateur dans la définition des requêtes exprimant ses intentions.
MRDSM permet de définir des **attributs dynamiques**. Ce sont des attributs qui permettent d'établir la traduction de données conflictuelles dans leur représentation. Par exemple, la taille d'une personne dans la BD_1 est exprimée en mètre et dans la BD_2 en inch. ces attributs permettent la transposition dans une unité au choix, pouvant différé des unités originales (exemple: le centimètre). La définition de ces attributs est réalisée par l'intermédiaire du langage MDSL.

C. Architecture fédérée fortement couplée mono-schéma

La caractéristique majeure de cette architecture est la présence d'un schéma fédéré reprenant

l'ensemble des schémas exportés et défini par l'administrateur de la fédération.

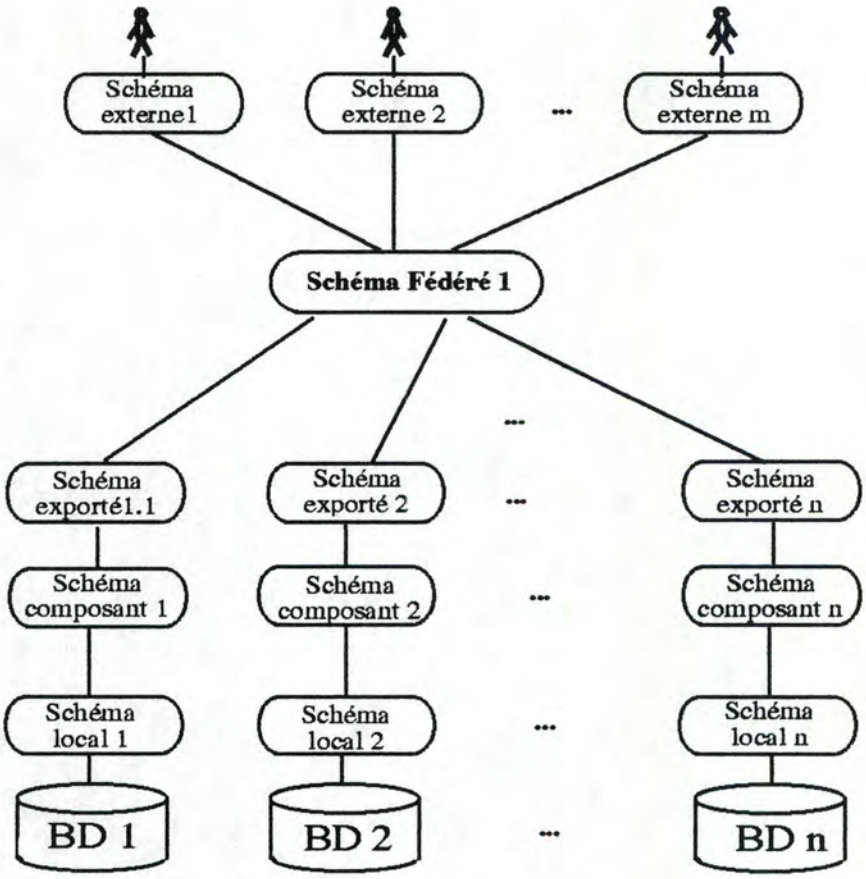


Figure 2.8.: Architecture fédérée fortement couplée mono-schéma

Le processus d'intégration des schémas exportés en un schéma fédéré se déroule en trois étapes:

1. Négociation entre l'administrateur de la future fédération et les administrateurs des bases de données locales pour la définition des schémas exportés. L'administrateur de la base de données fédérée est donc autorisé à consulter les schémas composants dans leur entièreté afin de prendre connaissance de leur contenu.

2. Création du schéma fédéré par l'administrateur de la fédération qui en assurera également la maintenance. En général, l'intégration est réalisée automatiquement ou semi-automatiquement.

3. Négociation entre l'administrateur de la fédération et les utilisateurs pour définir les schémas externes.

Tout SBDF répondant à cette architecture est qualifié de **statique** vu que le processus d'intégration est réalisé automatiquement sous le contrôle de l'administrateur de la fédération. Ce

type d'intégration est déterministe en ce sens que l'application du processeur de traitement à des schémas identiques fournira toujours le même schéma intégré.
L'avantage de cette approche est qu'elle fournit un seul schéma intégré, complet, c'est-à-dire qu'il reprend l'ensemble des éléments des divers schémas exportés et n'offre donc qu'une seule sémantique. Cette architecture rend donc la gestion des mise-à-jour beaucoup plus facile vu l'unicité de la sémantique.
Par contre, cette solution présente un inconvénient, celui de l'intégration. Il est en effet très difficile de définir un schéma complètement intégré, vu les ambiguïtés présentes dans les schémas et les conflits existants entre les différents schémas.
Remarquons le cas particulier où tous les SBD locaux mettent à disposition l'intégralité de leurs schéma conceptuel et non plus une partie de celui-ci. Cela reviendrait à une situation analogue à l'approche intégrée.
On peut résumer cette architecture par l'expression **architecture "read-write, mono-schema, non-oriented user".**

C.1. Etude de cas: DDTS [SHETH 90]

L'architecture de DDTS (figure 2.9.) se présente selon deux groupes de processeurs: l'unique processeur d'application et de multiples processeurs de données. Cette distinction est basée sur le fait qu'un seul schéma fédéré est défini et que plusieurs bases de données locales sont admises. On aura donc autant de processeurs de données que de systèmes de bases données dans la fédération.

Le modèle canonique utilisé par DDTS au niveau composant et fédéré est le modèle relationnel auquel est associé le langage de manipulation SQL. On constate qu'au niveau externe, un des modèle de données employé est un modèle riche en sémantique, le modèle Entity-Category-Relationship (ECR), défini par Elmasri, et auquel est associé le langage de requête GORDAS. L'autre modèle est le modèle relationnel et le langage SQL. Cela montre le désir des concepteurs d'offrir une architecture multi-interfaces. En effet, cette caractéristique permet de toucher deux catégories d'utilisateurs et diminue ainsi la tâche d'apprentissage d'un nouveau langage de manipulation, mais cela impose évidemment de traduire les requêtes de type GORDAS en requêtes conformes à SQL et au schéma global. Nous remarquons que cette phase de traduction est effectuée en une seule étape contrairement à CARNOT qui faisait cette traduction en deux étapes, une syntaxique et une sémantique.

Cette architecture ne présente pas de schéma exporté; autrement dit, elle néglige le désir d'autonomie d'association des bases de données locales. Il est toutefois à remarquer que le schéma local offert à la fédération pourrait tout simplement être une vue externe de ce schéma pour autant que le SGBD local reconnaît ce mécanisme dans sa totalité tout comme SQL le fait.

A présent examinons la fonctionnalité de chacun des processeurs présenté dans l'architecture.

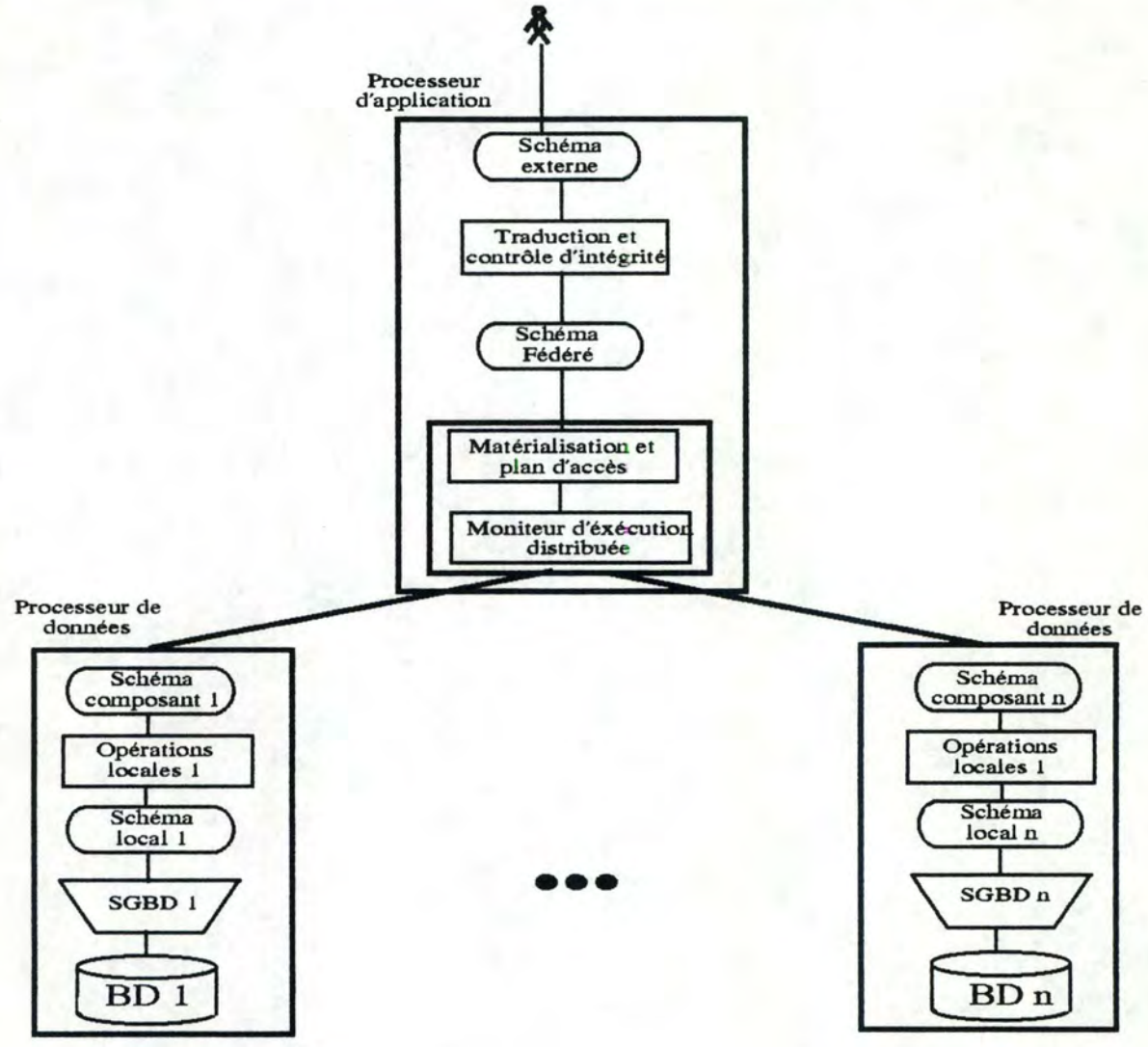


Figure 2.8. : Architecture de DDTS

Tableau 2.3

Référence	DDTS	Modèle	LMD
Sch. Local	Sch. Local Interne	Modèle quelconque	Langage quelconque
Sch. Composant	Sch. Local de représentation	Modèle relationnel	Langage SQL
Sch. Fédéré	Sch. Global de représentation	Modèle relationnel	Langage SQL
Sch. Externe	Sch. Conceptuel	Modèle ECR, Modèle relationnel	Langage GORDAS, Langage SQL

Les processeurs de traitement

Processeur de traduction et de contrôle d'intégrité: ce processeur traduit les requêtes GORDAS

formulées sur le schéma externe en requêtes relationnelles (SQL) exprimées sur le schéma fédéré et vérifie la requête pour qu'elle ne viole pas les contraintes d'intégrité lors de l'exécution, au besoin il l'a modifie. Si le schéma externe est de type relationnel, le processeur de traduction est naturellement absent.
Processeur de matérialisation et de définition du plan d'accès: ce processeur s'occupe de l'optimisation des requêtes globales. A cette fin, il définit une stratégie d'exécution distribuée consistant en un ensemble de requêtes, chacune étant exprimée en terme des schémas composants. Ce processeur offre également de sauvegarder la requête pour une éventuelle utilisation ultérieure.

Processeur de monitoring d'exécution: ce processeur est responsable de l'exécution de la stratégie d'exécution distribuée et coordonne les résultats locaux pour rendre un résultat unique à l'utilisateur.

Processeur des opérations locales: ce processeur est chargé de traduire les requêtes exprimées sur les schémas composants en requêtes formulées sur les schémas locaux et dans le langage du SBD local telles qu'elles puissent être exécutées par le SGBD local.

D. L'architecture fédérée fortement couplée multi-schémas

Cette architecture mêle les deux précédentes en admettant plusieurs schémas fédérés mais devant être définis par un ou plusieurs administrateurs de fédération.
On retrouve ici une multiplicité de schémas et donc de sémantiques ayant pour corollaire le problème des mise-à-jour comme il a été exposé auparavant. Mais étant donné que les schémas sont définis par des personnes spécialisées (les Administrateurs de Bases de Données), ces dernières peuvent définir les règles de mappings telles que les opérations de mise-à-jour respectent l'intégrité des BD locales. Si ces règles ne sont pas établies correctement, on devra faire face à une croissance des inconsistances des données des BD locales.

D.1. Etude de cas: MULTIBASE [SHETH 90, THOMAS 90]

Il s'agit d'une architecture développée par Landers et Rosenberg et exposée à la figure 2.9.

Les requêtes sont exprimées sur le schéma fédéré-externe dans le langage DAPLEX. Un seul langage et un seul modèle(fonctionnel) sont offerts aux utilisateurs de MULTIBASE. Ce système impose donc l'apprentissage de ce langage. Celui-ci est, jusqu'à présent, limité à des opérations de consultations. Les mise-à-jour ne sont pas traitées au niveau global pour des raisons évidentes de respect d'intégrité des BD locales. Elles sont toutefois permises au niveau local et de façon individuelle.
Aucun changement local ne doit être réalisé pour implémenter Multibase. Il est limité aux modèles relationnel et Codasyl comme systèmes locaux mais peut offrir des extensions.
Trois types de composants se distinguent:

- le **gestionnaire de données global** (Global Data Manager, **GDM**): composant central, il gère la manipulation des requêtes globales, depuis la traduction jusqu'à la coordination de l'exécution distribuée en passant par l'optimisation, la décomposition et le filtrage des requêtes globales.

- les **interfaces de bases de données locales** (Local Database Interfaces, **LDI**): chaque LDI reçoit du GDM des sous-requêtes exprimées dans le langage DAPLEX. Le LDI se charge de traduire ces requêtes conformément au schéma et au langage du SGBD local avec lequel il communique. Il fait en sorte d'optimiser l'exécution ultérieure des requêtes si les SGBD local ne le fait pas. Ensuite, il transmet la requête à exécuter au SBD interne auquel il est attaché.

- les **systèmes de gestion de BD interne** (Internal DataBase Systems)

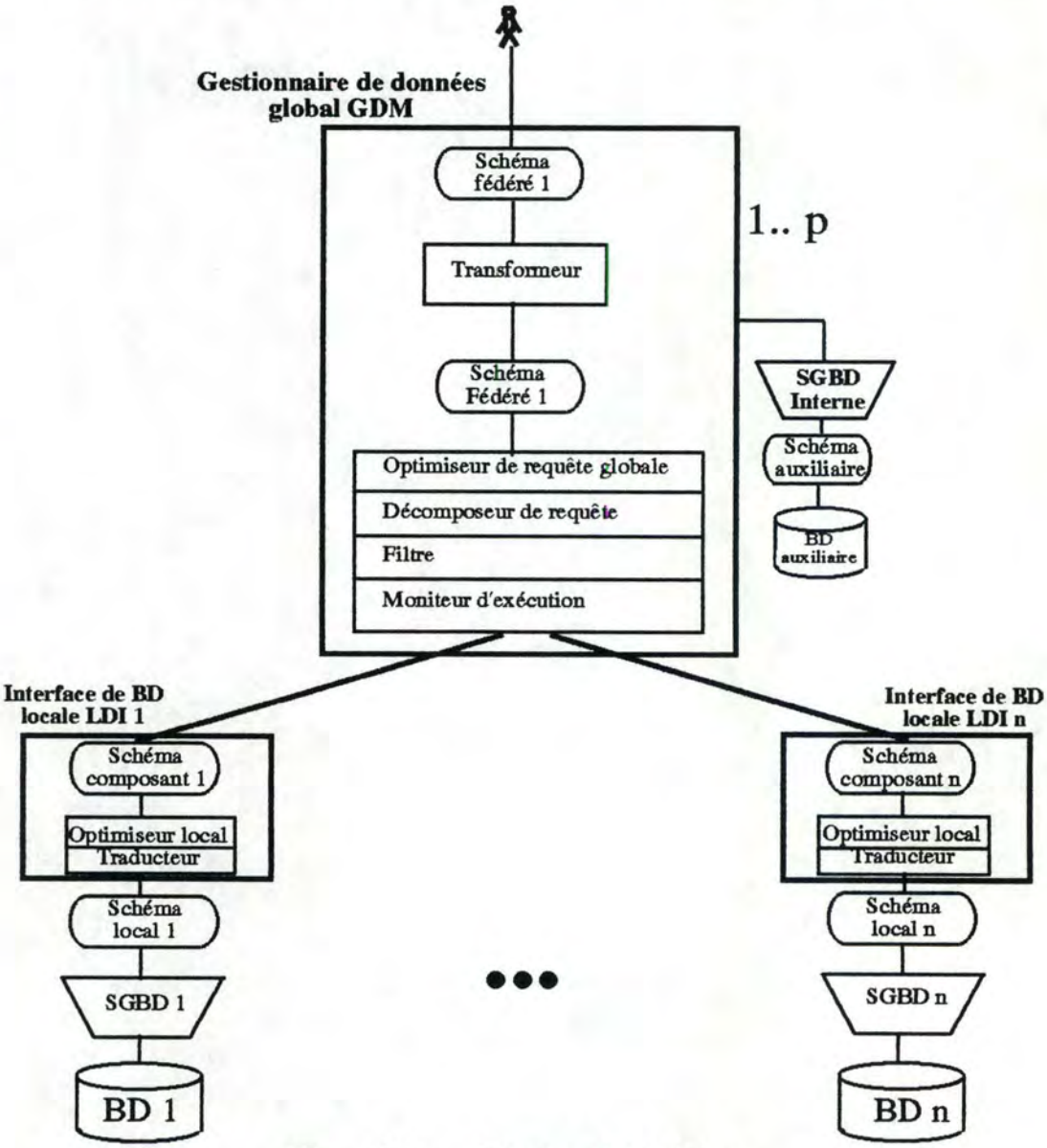


Figure 2.9.: architecture de Multibase

Le tableau 2.4. fournit les correspondances entre la terminologie de référence et celle de Multibase.

Tableau 2.4

Référence	MULTIBASE	Modèle	LMD
Sch. Local	Sch. Local	Codasyl, relationnel	Codazyl DML,SQL
Sch. Composant	Sch. Local Daplex	Fonctionnel Daplex	Langage Daplex
Sch. Fédéré	Vue Globale Daplex exprimée en termes locaux	Fonctionnel Daplex	Langage Daplex
Sch. Fédéré-Externe	Sch. Conceptuel	Fonctionnel Daplex	Langage Daplex

Face à cette architecture, on constate deux niveaux de schémas fédérés, l'un exprimant le schéma global intégré et l'autre exprimant ce même schéma en termes locaux, c'est-à-dire en faisant référence aux schémas locaux et au schéma auxilaire.
Le **schéma auxiliaire** regroupe des informations non disponibles dans les SGBD locaux et des informations nécessaires (tables de conversion, statistiques...) pour résoudre les inconsistances de données telles que les différences de noms, de structures, d'échelle, les données manquantes et les données conflictuelles. Ce schéma peut être interrogé grâce à l'utilisation d'un SGBD interne.

Les processeurs de traitement

Le *transformeur* modifie les requêtes globales DAPLEX en insérant des références vers les schémas locaux et auxiliaires.

Les quatre processeurs suivants s'attachent à générer une séquence de requêtes DAPLEX destinées chacune à un seul SBD local:

L'*optimiseur de requête globale*: il produit un plan global d'optimisation. L'optimum est calculé en fonction du volume de données transmises entre sites et du degré de parallélisme dans le traitement des sous-requêtes

Le *décomposeur de requête*: il décompose le plan global en requêtes destinées à un seul site.

Le *filtre*: il réduit les requêtes décomposées en enlevant les opérations qui ne sont pas fournies par le SBD local. C'est le SGBD interne qui se chargera de fournir ces opérations.

Le *moniteur*: il contrôle l'exécution distribuée des sous-requêtes.

L'*optimiseur local*: il détermine la stratégie de traitement optimal de la requête locale. Cette stratégie consiste à minimiser le temps d'accès aux BD locales en utilisant au mieux les opérations locales (services du SGBD local) et l'organisation physique de la BD.

Le *traducteur*: il convertit la requête exprimée en DAPLEX en une requête exécutable par le SGBD local.

D.2. Etude de cas: MERMAID [TEMPLETON 87]

Ce système permet de faire face à trois types de SGBD relationnel différents. Deux types d'interface utilisateur sont disponibles: soit le langage ARIEL associé à un schéma sémantique, soit le langage SQL pour les schémas relationnels. Les schémas **externes** sont donc du type relationnel (A) ou sémantique (B).

Il faut remarquer l'utilisation d'un langage intermédiaire (Distributed Intermediate Langage) pour les communications internes. Ce langage est défini sur un modèle relationnel, modèle canonique de l'architecture, et est totalement indépendant des langages d'interrogation.

L'architecture accorde une autonomie totale aux SBD locaux en leur octroyant le droit de limiter le partage de leurs données et en permettant toujours les accès locaux indépendamment du système global. Par contre, Mermaid se contente de traiter uniquement les opérations de consultation, préférant que les mise-à-jour soient directement définies au niveau local. Les opérations de consultation sont réalisées en mode interactif, c'est pourquoi beaucoup d'importance est accordée au temps de réponse du système et donc à la stratégie d'optimisation.

La figure 2.10. expose l'architecture globale de Mermaid et le tableau 2.5. les correspondances de terminologies.

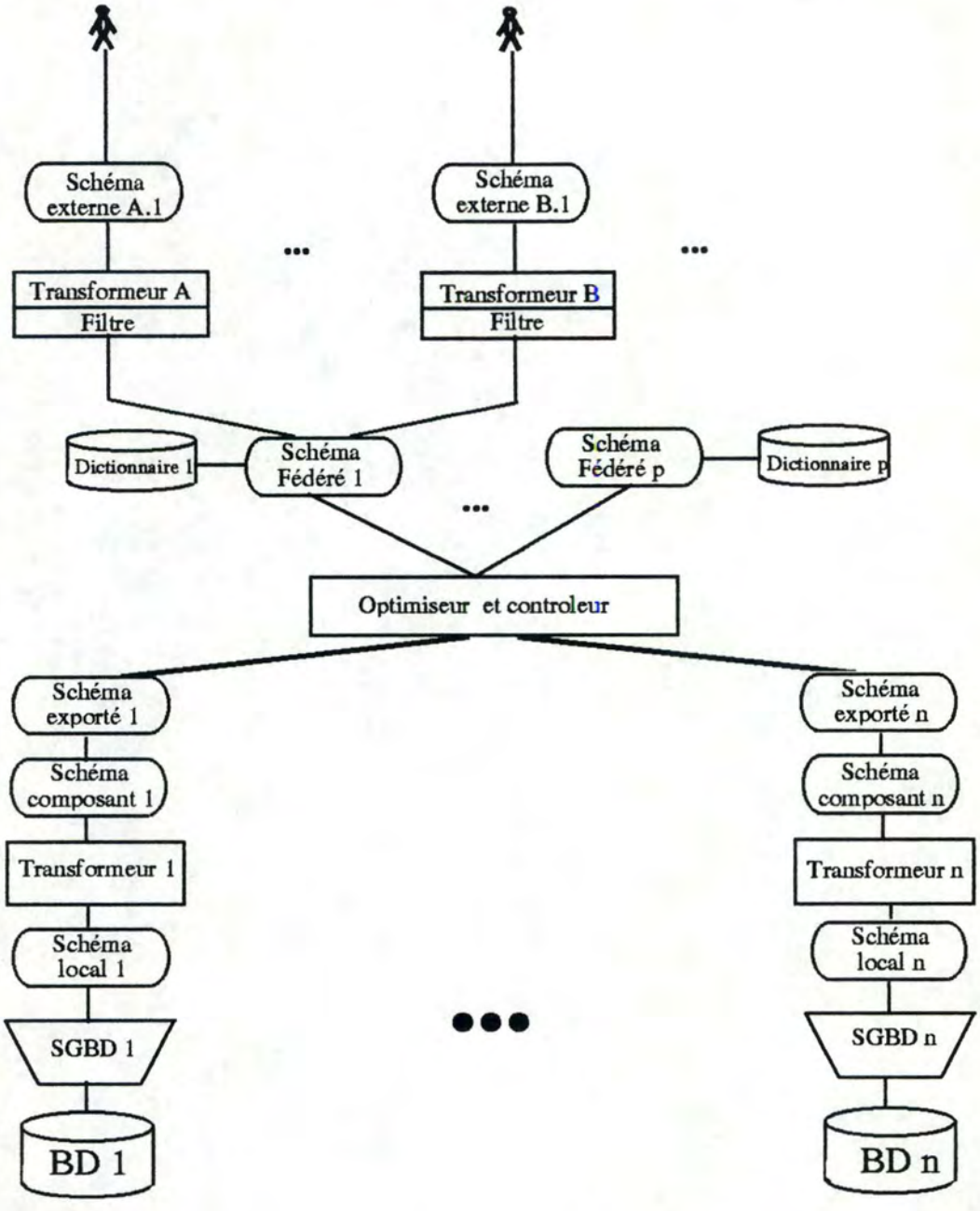


Figure 2.10. : Architecture de MERMAID

Les dictionnaires associés aux schémas fédérés contiennent des informations sur les BD locales, les utilisateurs, les ordinateurs et SGBD des SBD locaux, les types et performances des réseaux de communication, les caractéristiques physiques des données. Toutes ces informations sont essentielles pour la phase d'optimisation.

Tableau 2.5

Référence	MERMAID	Modèle	LMD
Sch. Local	Sch. Local	Ingres, IDM	Ingres'DML, IDM
Sch. Composant	Sch. Local Distribué	relationnel	DIL
Sch. Exporté	Vue externe	relationnel	DIL
Sch. Fédéré	Sch. Global	relationnel	DIL
Sch.Externe	Sous-schéma	relationnel, sémantique	SQL ARIEL

Les processeurs de traitement

Transformeurs: SQL - DIL, ARIEL - DIL: ces processeurs traduisent les requêtes exprimées en SQL ou ARIEL en langage DIL, hautement structuré. Ainsi il devrait être assez aisé d'étendre la gamme d'interfaces utilisateurs disponibles.
Afin de faciliter les étapes ultérieures, ces processeurs sont chargés de simplifier autant que possible les requêtes. Par exemple, les requêtes imbriquées sont décomposées en plusieurs requêtes.

Filtre: procède à un filtrage des accès

Optimiseur et contrôleur: Ce processeur définit et exécute un plan d'accès. L'optimisation est fonction de la rapidité des réseaux et des ordinateurs locaux. En fonction de ces paramètres, l'optimiseur essaie de minimiser le temps de réponse en minimisant les temps d'accès locaux et les temps de communication sur le réseau.

Transformeur 1..n: ces processeurs, un pour chaque SBD composant, traduisent les requêtes DIL en requêtes exécutables par le SGBD local. Ce même processeur se chargera de traduire dans le format relationnel intermédiaire les données qui lui seront fournies en retour et de les fournir au processeur responsable d'intégrer les données. Lors de la traduction des données, des algorithmes ou tables de correspondances sont utilisées afin de fournir des données exprimées uniformément.

D.3. Etude de cas: PEGASUS [AHMED 91]

PEGASUS est un système de gestion de bases de données hétérogènes dont une des caractéristiques principales est l'utilisation d'un modèle de donnée canonique de type orienté-objet (IRIS). Ce système envisage d'intégrer à la fois de simples fichiers textes jusqu'aux bases de données multimédia. L'autonomie des SBD locaux est tout à fait respectée dans ce système.

Le modèle orienté-objet de PEGASUS (IRIS) est défini avec les éléments suivants:

TYPE: représente une collection d'objets partageant les mêmes caractéristiques. Les TYPES sont organisés selon un graphe acyclique orienté qui supporte les mécanismes de généralisation, de spécialisation et d'héritage. Chaque TYPE a un nom unique

OBJET: identifié par un OID (Object IDentifier) unique, un objet appartient à un type. Un objet peut passer d'un type à un autre.

FONCTION: est la manifestation des opérations et fournit des correspondances entre objets. Les propriétés des objets, les relations entre objets et les calculs sur les objets sont exprimés en terme de fonctions.

Le langage de manipulation de données est également celui servant à leur définition. Il s'agit du langage HOSQL (Heterogeneous Object Structured Query Langage). Il a la propriété d'être **assertionnel.**

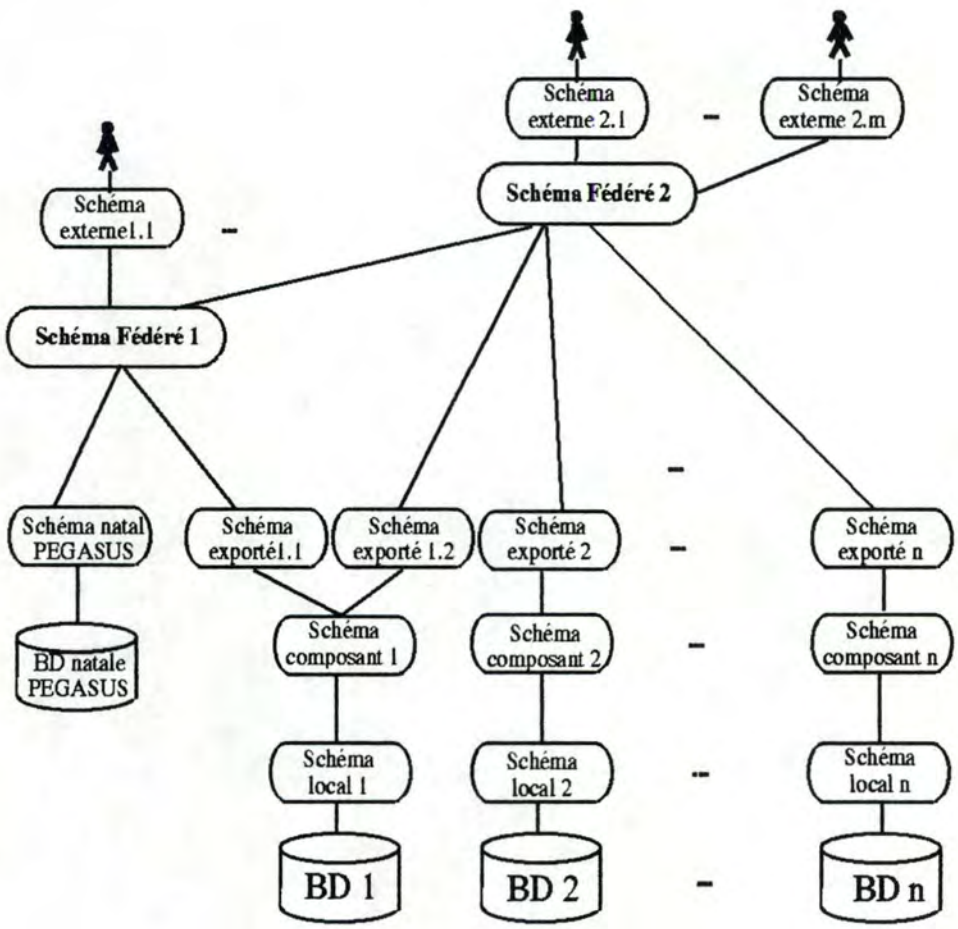


Figure 2.11. : Architecture de PEGASUS

Le tableau 2.6 présente les correspondances de terminologie entre l'architecture de référence et celle de PEGASUS.

Tableau 2.6

Référence	PEGASUS	Modèle	LMD
Sch. Local	Sch. Local	Quelconque	Quelconque
Sch. Composant	Sch. Composant	Orienté-objet IRIS	HOSQL
Sch. Exporté	Schéma Importé	Orienté-objet IRIS	HOSQL
Sch. Fédéré	Sch. Intégré	Orienté-objet IRIS	HOSQL

L'architecture de schéma reprend un schéma appelé **schéma natal**. Ce dernier est en réalité le schéma d'une base de données PEGASUS existante. Les schémas des bases de données locales viennent s'intégrer au schéma natal afin d'accéder aux SBD locaux par l'intermédiaire de PEGASUS. On constate la possibilité d'intégration de plusieurs schémas locaux avec un schéma fédéré pour créer un autre schéma fédéré. C'est une caractéristique intéressante mais remarquons que les autres architectures analysées n'interdisent nullement que l'un des SBD local soit déjà un SBDF.

La phase d'intégration se fait à l'aide d'un outil semi-automatique dirigé par l'administrateur de la BD fédérée. Cet outil travaille au départ de commandes exprimées en HOSQL et déduit automatiquement les règles de mapping. Avant toute intégration d'une base de données, il faut fournir à PEGASUS les paramètres techniques de cette base: SGBD, protocole réseau, adresse du noeud sur le réseau, type de machine,... La technique d'intégration consiste essentiellement en la création de super-types des types définis dans les BD sous-jacentes. La résolution des conflits de domaines se fait pendant le définition des règles de mapping.
PEGASUS permet d'interroger le schéma exporté directement sans construire de schéma global mais uniquement de façon individuelle.

Dans l'architecture complète de ce système, on retrouve des processeurs de décomposition de requêtes, d'optimiseur, de traduction de schémas et de requêtes tout comme dans les architectures précédantes.

L'objectif de cet état de l'art était plus orienté vers les approches de résolution que vers les techniques utilisées pour résoudre les problèmes particuliers. De plus l'ensemble des études de cas présentées constituent un éventail restreint des architectures existantes. Pour des raisons évidentes il était impossible de présenter en détail le fonctionnement de chaque système. On peut trouver d'autres systèmes, telles que ADDS, CALIDA, OMNIBASE, HD-DBMS, IMDAS, DATAPLEX.

Nous allons dès lors critiquer ces approches (Chapitre 3) et ensuite analyser plus précisément les

différents problèmes sous-jacents à la gestion de BDH (Chapitre 4) et qui ont été mis en évidence par cet état de l'art.

CHAPITRE 3: CRITIQUE DES APPROCHES

Ce chapitre a pour objectif de faire ressortir les avantages et inconvénients des approches en regard de notre objectif de triple transparence mais également des domaines dans lesquels on peut appliquer ces approches. A l'issue de cette critique, nous choisirons une des architectures pour développer plus amplement les problèmes sous-jacents à l'environnement des BDH.
Les angles de vue selon lesquels nous critiquerons les approches sont incontestablement les transparences de localisation/distribution et d'hétérogénéité ainsi que le degré d'autonomie accordé aux systèmes locaux, ce dernier étant étroitement lié à définition du schéma résultant de l'intégration.

L'**approche intégrée** présente l'avantage d'offrir un seul schéma global intégré complet. Ainsi, **une seule sémantique** de l'ensemble des éléments des différents schémas est adoptée. Cela permet d'offrir aux utilisateurs une **vue unique** sur l'ensemble des bases de données locales. La définition du schéma global, caractérisé par son uniformité et sa complétude, rend la gestion des **opérations de modifications** plus facile. En effet, vu qu'une seule vision des concepts est reconnue, la propagation des opérations de mise-à-jour se fera sans conteste sur base des règles de mappings établies lors de la phase d'intégration de schémas. La définition de **schémas externes** sur base du schéma global permet aux utilisateurs de sélectionner les parties de schéma d'intérêts pour leurs usages personnels. De plus la possibilité de travailler sur des schémas externes décrits dans un formalisme étranger au modèle canonique peut faciliter la tâche des utilisateurs en conservant leurs habitudes.
De plus, l'intégration se faisant sur base des schémas conceptuels transposés dans le modèle canonique oblige les systèmes locaux à **abandonner leur autonomie**. En effet les systèmes locaux n'ont pas le pouvoir de négocier le partage de leurs schémas. De plus cette approche n'autorise pas la **distinction entre requête locale et globale**. Ainsi il devient impossible aux utilisateurs des SBD locaux d'interroger leurs bases indépendamment du système global et de son schéma global. Cette limitation peut constituer un avantage dans la mesure où elle permet d'annuler la croissance des données incompatibles. En effet, la modification individuelle des bases de données locales peut mener à des mises-à-jour incompatibles. Ces dernières ont des origines multiples: oubli de modifier une des bases, frappe erronée, dé-synchronisation des opérations,... Donc si l'ensemble des bases de données locales est compatible avant la constitution du SBDH, il le restera toujours. De même si l'ensemble des bases de données locales est incompatible à 10 %, la définition du SBDH permettra de conserver le taux d'incompatibilité, voire de le diminuer.
Il faut aussi remarquer que la conception du schéma global intégré est une **tâche extrêmement complexe** vu la multitude des conflits pouvant exister entre les schémas. De nombreux auteurs ont travaillé et travaillent encore sur ce problème. A cela s'ajoute le problème d'**extensibilité** de l'éventail des SBD locaux. En effet, lors de l'adjonction ou de la modification d'un SBD local, le processus d'intégration est à refaire afin de ré-uniformiser la vue globale et l'ensemble des règles de mapping. Les études de cas proposées ont montré qu'il était possible de remédier à ce problème en adoptant une méthodologie d'intégration spécifique.

Cette approche convient donc mieux à une situation où on préfère que chaque utilisateur ait une une vue identique de l'ensemble des données, permettant ainsi une grande diversité d'opérations de consultation et de mise-à-jour. On y souhaite également un partage intégral de l'ensemble des données des BD locales et donc une mise à la disposition de tous les utilisateurs de ces données. L'accès pourra toutefois être limité au niveau des schémas externes. Cette situation privilégie les accès par le schéma global au détriment des accès locaux individuels pour des raisons de conservation d'intégrité tant individuelle que collective des BD locales.

L'**approche multi-bases-de-données** a été très peu développée vu son faible degré de résolution du problème des BDH. Elle offre toutefois l'avantage de ne pas avoir de schéma global intégré complet ce qui élimine le problème d'intégration global. En effet, l'accès aux multiples bases de données est réalisé à l'aide d'un langage multi-bases qui permet de manipuler plusieurs bases de données simultanément. Cette solution induit que l'utilisateur doit connaître la localisation des données et la définition du schéma composant de chaque base de données local. Par contre, cette technique laisse aux systèmes locaux leur autonomie complète.
A une requête multi-bases, le système répond par une réponse intégrée; il y a donc intégration dynamique. Cette intégration porte uniquement sur les parties de schémas pertinentes à la requête. Rappelons que la requête contient le nom de chaque base de données et celui de l'élément souhaité. Cela a l'inconvénient de ne pas cacher la localisation réelle des données. Cette approche est recommandée pour des accès temporaires à une collection de base de données. Le temps d'accès y sera plus long mais le temps de développement du système sera plus court. Ce rapport est largement accepté si l'on ne fait que des accès peu fréquent à une grande collection de base de données. En effet pourquoi intégrer plusieurs centaines de schémas de bases de données, voire plus encore si ce n'est que pour un accès tous les 6 mois.
Ce type d'approche permet à la fois les opérations de consultation et de mise-à-jour mais l'inconvénient réside dans l'impossibilité de respecter les règles d'intégrité collective.
L'extension de la collection des SBD locaux n'est pas limitée vu l'absence de schéma global intégré.

L'**approche fédérée** permet de bénéficier d'une transparence de distribution et d'une autonomie partielle des systèmes locaux. En effet la définition d'un niveau *"schéma exporté"* permet aux systèmes locaux de limiter le partage de leurs schémas et ainsi de garder un contrôle d'accès à leurs données locales. Ce procédé de limitation des données partagées constitue une norme de sécurité très intéressante et facile à mettre en oeuvre. La technique du schéma fédéré offre un schéma global, défini sur les schémas exportés, et donc facilite l'accès en retirant à l'utilisateur le problème de la localisation des données. On retrouve ici, par analogie avec l'approche intégrée, le problème de définition d'un ou plusieurs schémas "globaux" . Ce problème limite l'extension du SBDH vu la nécessité de mettre à jour le(s) schéma(s) fédére(s) lors d'une modification de l'éventail des SBD locaux.
La multiplicité des schémas fédérés permet d'accepter plusieurs sémantiques aux schémas locaux en fonction des utilisateurs. Cette particularité est avantageuse mais peut s'avérer être un

inconvénient pour la gestion de certaines opérations telles que les mises-à-jour. C'est pourquoi nous allons discuter des trois types de systèmes fédérés séparément.

L'architecture fédérée ***faiblement couplée*** laisse aux utilisateurs le soin de définir le schéma fédéré et de le gérer. L'utilisateur est responsable du schéma fédéré depuis sa définition jusqu'à sa destruction. Il doit donc choisir les schémas qui lui sont utiles et résoudre les problèmes de conflit entre ces schémas quels qu'ils soient afin de définir le schéma intégré. Généralement, la phase d'intégration se fait à l'aide d'un langage d'intégration-restructuration tel que celui par [MOTRO 87]. Cette architecture admet implicitement une multitude de schémas fédérés vu les besoins différents des utilisateurs et pose donc le problème de la gestion des mise-à-jour. En effet, il est difficile d'assurer une intégrité des multiples base de données dans ces conditions. On a d'ailleurs qualifié cette approche de "read-only approach". De ce fait, elle conviendrait parfaitement dans une situation où l'on souhaite limiter le partage des données de façon dictatoriale c'est-à-dire sans négociation entre fournisseurs et utilisateurs des données mais également limiter les opérations à de simples consultations. Les opérations de mises-à-jour seront toujours réalisées au niveau local avec comme conséquence possible une croissance des incompatibilités entre bases de données. Cette solution est réservée à une élite d'utilisateurs vu les connaissances requises pour effectuer l'intégration et la résolution des conflits entre schémas. Ces opérations nécessitent en effet la connaissance des schémas exportés, du formalisme utilisé par le modèle canonique et du langage d'intégration, ce qui n'est pas connu de tout utilisateur.

L'architecture fédérée ***fortement couplée mono-schéma*** implique moins l'utilisateur. En effet, seule l'administrateur de bases de données peut définir un schéma fédéré. Ce type d'architecture n'offre donc qu'une sémantique possible d'interprétation des schémas locaux. Cette phase d'intégration sera généralement réalisée de manière automatique ou semi-automatique. Cette approche rend plus aisée le problème de propagation des mise-à-jour sur les schémas locaux vu la caractéristique d'uniformité de conception. On peut rapprocher cette architecture de l'approche intégrée à laquelle on aurait ajouté un niveau dans l'architecture des schémas, ce qui permet aux systèmes locaux de conserver leur autonomie.
La définition des schémas exportés est réalisée par négociation entre administrateurs locaux et fédérés. De ce fait, le partage des données est plus démocratique et l'autonomie des systèmes locaux un peu affaiblie. Mais toutes les opérations continuent à être contrôlées au niveau local.
Cette architecture est celle qui conviendrait le mieux en entreprise. En effet tous les utilisateurs ont une vue uniforme de l'ensemble des données, chaque département de l'entreprise conserve le droit de négocier le partage de ses données et d'y accéder indépendamment du système global. La définition des schémas exportés est un excellent procédé de sécurité pour protéger ses données.

L'architecture fédérée ***fortement couplée multi-schémas*** constitue un mélange des deux approches fédérées précédentes. En effet, elle admet plusieurs schémas fédérés simultanément et ces derniers acceptent les opérations de modification vu que ceux sont les administrateurs de bases de données qui définissent ces schémas et sont donc capables de définir des règles de mapping adéquates pour assurer une cohérence individuelle et collective des bases de données locales. La négociation entre les administrateurs des fédérations et les utilisateurs est primordiale

pour définir des schémas fédérés correspondants aux attentes des utilisateurs.

Dans la suite de ce mémoire, nous continuerons à travailler sur l'**approche fédéré fortement couplée mono-schéma**. Ce choix est basé sur l'importante prise en considération de la triple transparence (localisation, hétérogénéité et autonomie) et de l'uniformité de conception qu'offre cette architecture. Cette approche est celle qui sera la plus convoitée lors de l'implantation en entreprise si les utilisateurs sont amenés à faire de nombreuses opérations de mise-à-jour.
Nous avons donc opté pour un contexte d'entreprise avec un nombre raisonnable de SBD et des utilisateurs n'appartenant pas une classe d'élites en informatique. L'administrateur de la bases de donnée fédérée aura à assumer de grandes fonctions de gestion mais également de négociation à la fois avec les administrateurs (fournisseurs) des SBD locaux mais également avec les utilisateurs (clients) afin de concilier les demandes des clients et les offres des fournisseurs.

La figure 3.1. rappelle l'architecture des schémas de l'approche fédéré fortement couplée mono-schéma

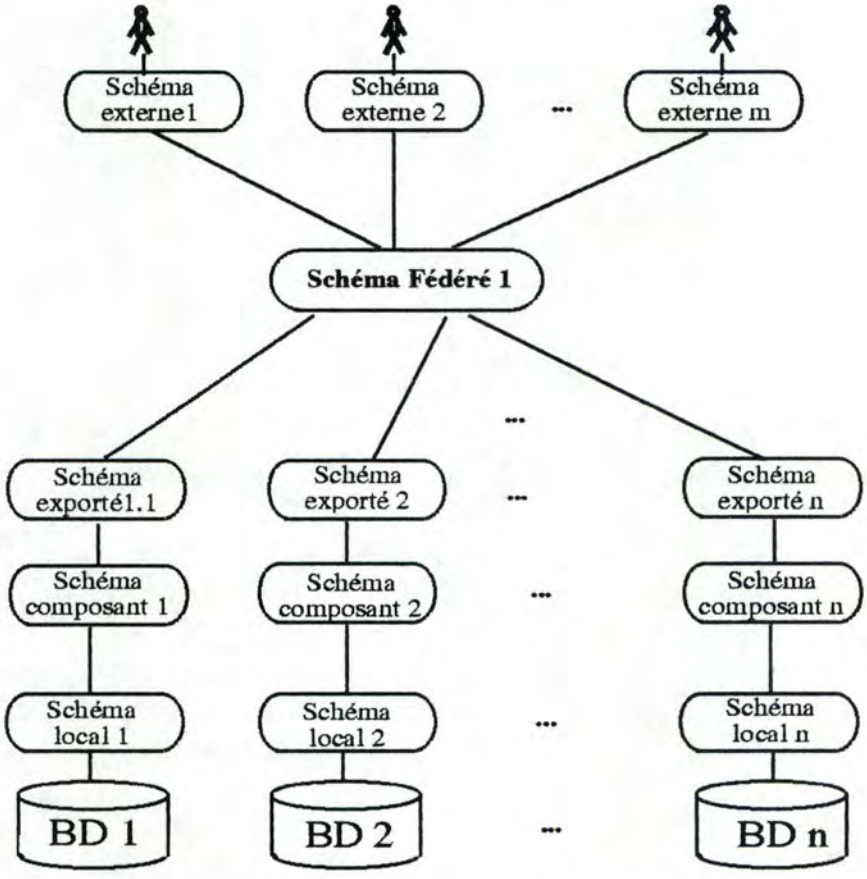


Figure 3.1. : Architecture fédéré fortement couplée mono schéma

CHAPITRE 4: SYNTHESE DES PROBLEMES EN BASES DE DONNEES HETEROGENES

L'état de l'art nous a permis d'identifier les problèmes à résoudre pour développer un système de bases de données hétérogènes. Bien qu'il y ait de nombreux problèmes à résoudre, on pourra constater qu'ils peuvent être regroupés en classes. Une classe reprendra des problèmes de même type. Ainsi, on établira une classification des problèmes; cette dernière sera illustrée par un exemple de BDH (figure 4.1). Les sections suivantes seront consacrées à l'analyse de chacune des classes de problèmes.

4.1. Classification des problèmes

Tout d'abord, insistons sur un problème particulier qui est celui du **choix du modèle canonique**. En effet, afin d'offrir une représentation uniforme de l'ensemble des schémas des bases de données réelles, on se doit de choisir un modèle unique et un langage (LMD et LDD) qui y est associé. Ce problème (**classe 1**) sera analysé à la section 4.2. On peut dès lors identifier quatre autres classes de problèmes.

Classe 2: Les problèmes de transformation

Sous cette classe, on retrouve les problèmes de transformation de schémas, de requêtes et de données. De manière générale, toute **transformation** consiste, à partir d'une ***structure source*** à produire une ***structure cible***, tout en vérifiant certaines propriétés. Le terme de ***structure*** est un terme **générique** pour les termes ***schéma***, ***requête*** et ***données***. Nous donnerons des définitions plus spécifiques lors de l'analyse de cette classe de problèmes (section 4.3).

Classe 3: Les problèmes d'intégration

Cette classe regroupe le problème d'intégration de schémas et celui d'intégration de données. De manière générale, l'intégration est un processus qui consiste à fournir une ***structure unique*** à partir d'un ***ensemble de structures***. Le terme de ***structure*** est **générique** pour les termes ***schéma*** et ***données***. Nous fournirons également des définitions plus précises dans la section 4.4.

Classe 4: Le problème de traitement des requêtes globales

Ce problème consiste à partir d'***une requête*** (requête globale) définie sur le schéma global intégré (ou fédéré) à déterminer un ***ensemble de sous-requêtes*** vérifiant certaines propriétés avec la requête globale. Etant donné que le schéma global est un schéma virtuel, c'est-à-dire ne présentant pas de données réelles, cette décomposition en sous-requêtes est indispensable pour interroger les bases de données réelles.

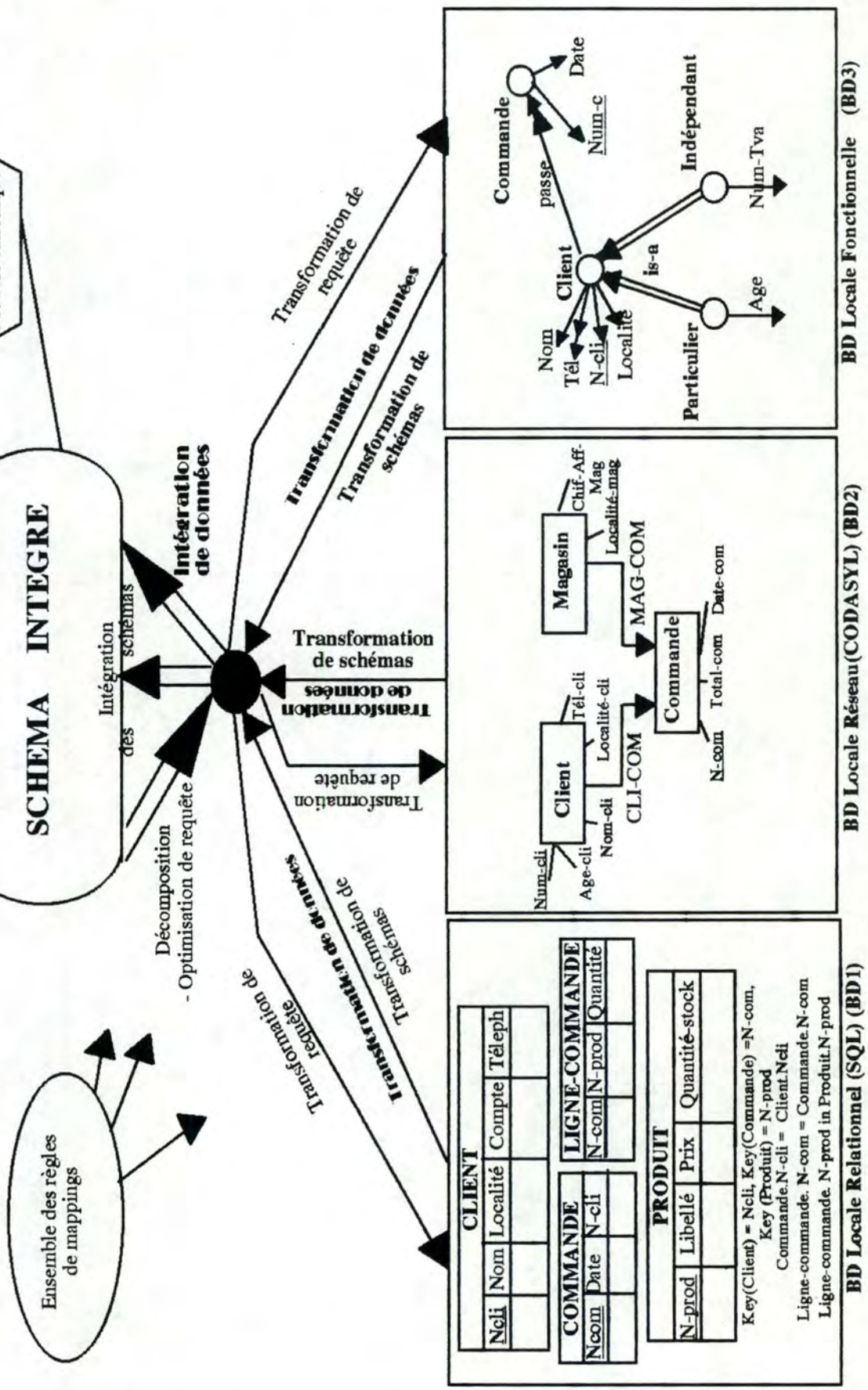


Figure 4.1. : Exemple de bases de données hétérogènes

Le problème de décomposition aurait pu être classé avec les problèmes de transformations. Nous ne l'avons pas fait car il s'agit ici de transformer une requête en un ensemble de sous-requêtes. On a donc ***une structure source*** et ***n structures cibles*** pour reprendre les termes de la classe 2, contrairement aux problèmes de transformation classique qui ont ***une structure source*** et ***une structure cible.***

Nous avons vu que l'on doit faire appel aux services de nombreux SGBD pour résoudre une requête globale. Pour interroger ces bases de données réelles, nous devons utiliser des moyens de communication, tels que les réseaux locaux ou autres. Il importe d'envisager un traitement optimisé des requêtes globales afin d'offrir un temps de réponse acceptable. Ce problème étant étroitement lié à celui de la décomposition de requêtes, nous avons décidé de les regrouper.

Nous analyserons ces problèmes à la section 4.5.

4.2. Le choix du modèle canonique

Face à l'hétérogénéité des modèles des SBD locaux, une mesure s'impose d'emblée pour assurer une certaine homogénéité. A cette fin, il convient de choisir un modèle de données, appelé **modèle canonique**, dans lequel on va traduire les schémas des SBD locaux. Vu le nombre de modèles existants et leurs propriétés discriminatoires, il est difficile de trancher le débat sans procéder à une analyse des qualités que doit offrir ce type de modèle. Il est également de rigueur de choisir un langage de manipulation de données associé à ce modèle.

On commencera par rappeler la fonction d'un modèle de données. On exposera alors l'importance de ce choix dans l'élaboration de la solution et par conséquent les caractéristiques générales que doit offrir ce modèle. Ensuite on définira une liste de critères que ce modèle devra vérifier. Sur base de cette grille d'analyse, on proposera une critique des modèles existants actuellement. Cela nous permettra d'éditer un choix de modèle canonique. Pour terminer, on changera d'optique en se tournant vers une solution faisant appel à un méta-modèle.

4.2.1. Fonction d'un modèle

De manière générale, un modèle permet de représenter tout ou partie du monde réel. La fonction d'un modèle est donc celle de **représentation du monde réel**. A cette fin, tout modèle propose des structures ou concepts et des règles ou opérations permettant à un individu d'exprimer sa conceptualisation du monde réel. Il est important de constater que chaque individu peut avoir une vue différente du monde réel et donc proposer une conceptualisation différente. A titre d'exemple, un garagiste ou une compagnie d'assurance de voitures n'ont pas la même conceptualisation d'une voiture. Le premier la décrit par la marque, la cylindrée, la couleur, la carburant utilisé, la vitesse maximale, le nombre de rapport de la boîte de vitesse, le nombre de cylindres, de soupapes, son prix,... Tandis que la compagnie d'assurance la définit par sa cylindrée, son numéro de châssis, sa puissance en kW, sa marque, son type. Même si ces deux personnes vivent grâce à la voiture, elles en ont des vues différentes.

On peut donc définir les caractéristiques d'un modèle en partant de sa **capacité de représentation** du monde réel. Cette dernière est examinée selon **deux dimensions**: l'expressivité du modèle et son relativisme sémantique.

L'expressivité donne le degré jusqu'au quel le modèle peut représenter une conceptualisation, quel qu'en soit sa complexité et ses concepts composants. Jusqu'il y a peu ces modèles se contentaient de représenter les aspects statiques de la conceptualisation c'est-à-dire les invariants de structure du système réel. Aujourd'hui, on étend cette représentation aux aspects dynamiques de la représentation c'est-à-dire au comportement des éléments du système réel. Dès lors la dimension d'expressivité des modèles se présente sous deux angles: l'angle de la **structure** et celui du **comportement**. L'expressivité analysée selon l'angle de la structure reflète le pouvoir des structures du modèle à représenter les concepts et à être interprété comme ces concepts tandis que l'autre reflète le pouvoir du modèle à représenter les comportements des composants. A titre d'exemple, on peut citer comme mécanismes d'expressivité structurale les structures de table, d'attributs,... pour le modèle relationnel; les structures d'entités, d'attributs, d'association, de contraintes d'intégrité statiques,... pour le modèle entité-association; les structures d'objets, de propriétés,... pour le modèle orienté-objets. Comme mécanisme d'expressivité comportementale, on citera les contraintes d'intégrité dynamique en entité-association; les méthodes ou opérations en orienté-objets.

Le relativisme sémantique d'un modèle désigne quant à lui la capacité du modèle à permettre différentes conceptualisations d'un même élément du monde réel(figure 4.2). Par exemple, dans le modèle entité-association, on ne peut représenter la situation "une entreprise fabrique plusieurs produits et un produit peut être fabriqué par plusieurs usines" que d'une seule façon: type d'entités(TE) USINE ,TE PRODUIT et un type d'association N-N entre ces deux TE. Par contre en orienté-objets, il existe plusieurs conceptualisations possibles: par exemple, les objets PRODUIT dans les objets USINE ou inversément.
Etant donné qu'une seule conceptualisation est exprimée dans le schéma d'une bases de données, cela limite la vision du monde réel à une vision unique. Or on souhaiterait parfois avoir une conception différente. C'est pourquoi on a défini le concept de schéma externe, qui représente la vue que l'utilisateur quidam a du monde réel. Par conséquent, on peut étendre la notion de relativisme sémantique en disant ceci: si le modèle permet de définir des schémas externes autres que des sous-schémas du schéma conceptuel, il a un degré élevé de ***relativisme sémantique externe***. Il semble en effet que la notion de relativisme sémantique et celle de mécanisme de définition de schéma externe sont fortement liées. Notons que si le modèle a un degré élévé de ***relativisme sémantique externe***, il a forcément un fort degré élevé de relativisme sémantique; le contraire n'est pas forcément vrai. Par exemple le modèle ERC+ [PARENT 87] a un degré élévé de relativisme sémantique mais un degré de ***relativisme sémantique externe*** faible puisque l'algèbre ERC+[PARENT 89] ne permet que de définir des sous-schémas du schéma conceptuel.

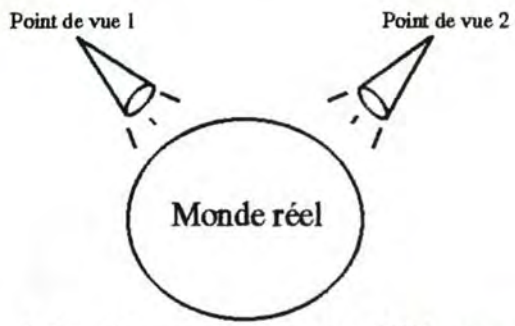


Figure 4.2. : Deux points de vue différents sur le monde réel menant à deux conceptualisations différentes

4.2.2. Importance de ce choix

Le choix de ce modèle est d'une importance cruciale pour la définition d'une architecture car celle-ci est totalement axée sur ce modèle. Il aura en effet un triple effet sur la solution: sur sa **flexibilité**, sa **complexité** et sa **facilité d'utilisation.**

Ce dernier influencera la **complexité** de réalisation de l'architecture. En effet un modèle simple, c'est-à-dire n'offrant que des mécanismes de représentation primitifs, risque de rendre la solution globale plus complexe alors qu'un modèle compliqué, c'est-à-dire offrant des structures de représentations puissantes, risque de rendre la solution globale plus simple. En effet, ce modèle servant de moyen d'homogénéisation des schémas locaux, s'il est trop simple, la phase de transformation de schémas sera d'autant plus complexe. Il s'avère en effet que la transformation de certaines structures complexes serait plus difficile à mettre oeuvre dans ce cas. Par contre, le choix d'un modèle offrant des mécanismes de structuration puissants rendrait cette tâche de transformation de schémas plus aisée, ce qui ne signifie pas qu'elle serait triviale. Mais un modèle de ce type peut poser des difficultés lors de la phase d'intégration de schémas. En effet, plus le nombre de structures est élevé, plus le nombre d'alternatives se présentant lors d'un conflit dû à l'hétérogénéité ***sémantique*** des schémas(Cfr typologie de la section 1.2 b) sera grand. Il s'agira donc de définir un ***ensemble de mécanismes de structuration bien équilibré.***

Ce modèle influencera également la **flexibilité** de la solution face à l'hétérogénéité potentielle des SBD locaux. Ce modèle devra entre autres être capable de représenter l'intégralité des schémas des BD locales dans son propre formalisme. De plus, si l'on souhaite étendre l'éventail du SBDH en lui annexant de nouveaux SBD locaux utilisant des modèles distincts de ceux utilisés par les systèmes déjà intégrés, il est indispensable que le modèle canonique utilisé puisse englober les modèles de ces nouveaux arrivants. Le non-respect de cette contrainte nous mènerait à une re-définition intégrale du SBDH en utilisant un modèle canonique plus adéquat. Par conséquent, le choix de ce modèle influe sur les perspectives d'extension du SBDH. Il est donc important que ce modèle soit désigné en gardant un regard sur les perspectives futures du systèmes de BDH.

Finalement, ce modèle déterminera la qualité potentielle des schémas externes, c'est-à-dire des

interfaces utilisateurs et donc la **facilité d'utilisation** du système. Un modèle ne permettant pas la définition de schémas externes ne constitue pas un frein majeur à son élection mais est néanmoins un atout de taille. On entend par là que cette possibilité permettrait de définir des schémas externes autrement qu'en définissant des procédures de transformation de schémas. Par exemple, le modèle relationnel permet grâce à son algèbre de définir des vues externes sans que l'on doive définir des procédures de transformation entre la vue et le schéma. Par conséquent, cela rend la solution également plus simple(effet sur la complexité de la solution).

On a ainsi montré l'importance incontournable du choix du modèle canonique. A présent on va établir une grille de critères permettant de rencontrer les remarques proposées ci-dessus.

4.2.3. Grille de critères

Face au triple effet de ce choix sur la solution, nous allons envisager de définir des critères permettant la sélection d'un modèle à la fonction de modèle canonique.

Afin d'assurer un rapport flexibilité/complexité acceptable, le modèle devra permettra la traduction de nombreux modèles en offrant une **gamme variée et complète de concepts de structuration.** Les mécanismes de représentation, encore appelés mécanismes d'abstraction puisqu'ils permettent d'avoir une vision abstraite de la réalité, devront au minimum être capable de modéliser tous les schémas des SBD locaux. Ceci implique que l'expressivité du modèle canonique devra être au moins égale à l'expressivité de chacun des modèles locaux. Dans des perspectives d'extensions futures, son expressivité devrait leur être supérieure tout comme pour la phase d'intégration. Cette dernière exige en effet des mécanismes d'abstraction puissants afin de faire face aux relations existantes entre les concepts des schémas. Ainsi, si on attribue à chaque modèle des SBD locaux un nombre représentant leur expressivité comme suit:

Degré-expressivité-modèle (SBD 1) = 2
Degré-expressivité-modèle (SBD 2) = 3
Degré-expressivité-modèle (SBD 3) = 1,

on souhaite que le modèle canonique ait un degré d'expressivité au moins égal à 3. Il est évident que la fonction "Degré-expressivité-modèle" est tout à fait artificielle. Il est en effet très difficile de définir une telle fonction. Nous avons supposé son existence afin de montrer la néccessité d'avoir un modèle canonique au moins aussi puissant que chacun des modèles des SBD locaux.
Cette recommandation reste néanmoins peu concrète. Nous allons donc préciser, plus concrètement cette fois, les mécanismes que doit offrir le modèle canonique.
Nous nous contenterons toutefois de considérer uniquement les mécanismes relatifs à l'expressivité structurale étant donné que les modèles intégrant à la fois l'expressivité structurale et comportementale commencent seulement leur percée dans le domaine de la conception de bases de données. Nous identifierons 7 critères qui nous semblent représentatifs pour le choix d'un modèle canonique pour un environnement de BDH.

Critère 1: Mécanisme d'instanciation - classification

Le modèle devra distinguer le concept de classe et d'instance. En effet tout modèle aujourd'hui propose cette distinction. Son exigence ne nécessite pas de commentaire supplémentaire si ce n'est la relation entre les deux: à une classe correspond 0, une ou plusieurs instances. Rappelons toutefois la distinction par l'exemple suivant:

Ex: **Classe** CLIENT(Nom, Prénom, Localité)
Instances (Dupont, Laurent, Namur)
(Durant, Pierre, Bruxelles)

Critère 2: Mécanisme d'agrégation - décomposition

Ce mécanisme est à la base de nombreux modèles. Il permet de voir une relation entre objets comme un objet autonome, comme un agrégat. Par exemple, la classe CLIENT est définie par **agrégation** des propriétés "Nom", "Prénom" et Localité". On dit que la classe client se **décompose** en les propriétés "Nom", "Prénom" et Localité".
Il existe plusieurs types d'agrégation. Nous reprendrons les deux plus utilisés: l'agrégation cartésienne et l'agrégation de couverture, encore appelée groupement.

L'**agrégation cartésienne** consiste à créer un objet comme produit cartésien d'autres objets. La flexibilité quant aux composants de l'agrégat permet de représenter des structures complexes. Par exemple, agrégat constitué des propriétés "Nom", "Prénom", "Adresse"; "Adresse", agrégat de "Rue", "Numéro", "Localité". Ainsi, on peut définir des **hiérarchies d'agrégats** , très utiles pour offrir des modélisations précises et notamment lors de l'intégration.

L'**agrégation de couverture** offre une représentation spécifique de groupe d'objets. L'exemple suivant illustre cette notion. Soient trois classes d'objets AVION, NAVIRE et HELICOPTERE. On construit la classe FLOTTE par agrégation de couverture des classes AVION, NAVIRE et HELICOPTERE. En effet, une instance de la classe FLOTTE est composée d'instances des classes AVION, NAVIRE et HELICOPTERE. Ce concept permet ainsi de regrouper des classes d'objets différentes.

Critère 3: Mécanisme de généralisation - spécialisation

La relation de généralisation - spécialisation entre deux classes A et B définit une relation

d'inclusion entre les les populations de ces classes. La figure 4.3 illustre des relations de cette sorte entre plusieurs classes. Ainsi on peut dire que tout "Motorisé" est "Véhicules". On désignera la première de ces classes comme classe **spécifique** et l'autre comme classe **générique.**

On peut ainsi raisonner à différents niveaux d'abstraction - niveau générique ou spécifique -, en fonction de nos besoins.

Un cas particulier de généralisation est celui de la **généralisation multiple:** à une classe spécifique correspond plusieurs classes génériques.

Ces deux mécanismes s'avèreront indispensables lors de l'intégration de schémas.

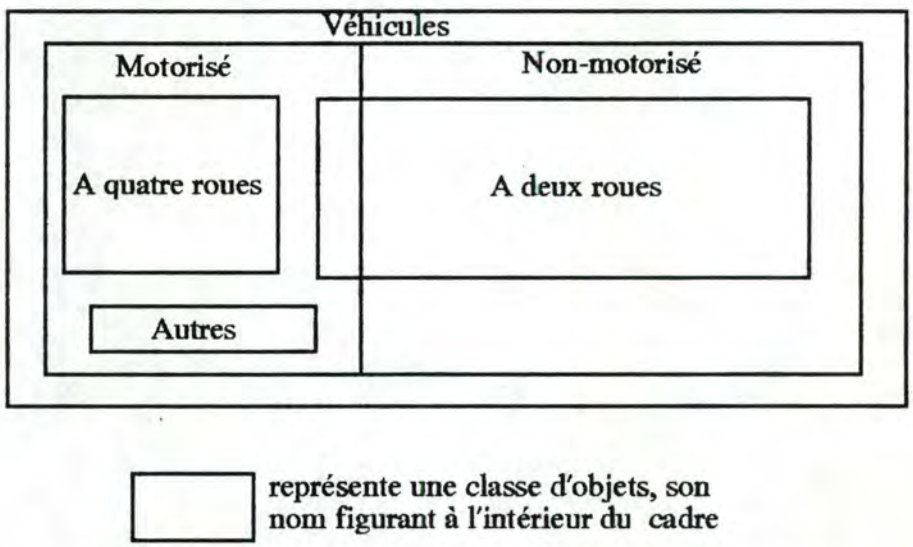


Figure 4.3. Relations d'inclusion entre populations de classes

Ainsi, le mécanisme de généralisation-spécialisation permet de voir un véhicule comme motorisé ou non-motorisé, à deux ou quatre roues,...Avec la complexité des relations d'inclusion et leurs propriété de transitivité, on définira une **hiérarchie de généralisations.**

Ce mécanisme est d'autant plus intéressant que lui est associé un puissant mécanisme d'inférence, celui de l'**héritage.** Ce dernier permet aux objets spécifiques d'hériter des propriétés du/des objets génériques correspondants. Cela permet donc d'alléger la structure des objets et de rendre le schéma plus lisible. Ainsi, les "à quatre roues" héritent des propriétés de "motorisé" qui hérite de "véhicule". Deux types d'héritages existent:

- de la classe générique vers la classe spécifique: **l'héritage descendant.**

- de la classe spécifique vers la classe générique: **l'héritage ascendant:** peut être utile mais est néanmoins peu utilisé. On ne considérera pas son absence comme un manque absolu.

On peut également associer au mécanisme de généralisation certaines propriétés qui permettent de préciser des relations entre les classes d'une hiérarchie de généralisation. Ce sont les

propriétés de **couverture** (tout véhicule est "motorisé" ou "non motorisé"), de **disjonction** (un "à deux roues" n'est pas un "à quatre roues") et de **partition** (disjonction et couverture en même temps). Ces propriétés permettent de préciser les relations entre les classes. Néanmoins, leur absence ne constituera pas une lacune.

Critère 4: Variété des domaines des propriétés des classes

Il est en effet souhaitable que le modèle offre une grande variété de domaines. Ceci suppose donc les **domaines de bases** (entier, réel, chaîne de caractères, booléen) mais surtout les **domaines construits** par les utilisateurs. En voici un très utile:

la **liste:** ex: propriété COULEUR: ("Rouge", "Vert", "Bleu", "Jaune")

Critère 5: Mécanismes d'affinement de la sémantique des classes

Lors de l'intégration, nous verrons que certaines propriétés ne sont pas reprises dans certains schémas. Cela nous conduira à rendre ces propriétés *facultatives* - c'est-à-dire pouvant ne pas présenter de valeur - dans le schéma intégré. Les propriétés des classes devraient donc souffrir de caractéristiques habituelles telles que: propriété **facultative ou obligatoire**, propriété **simple** ou **répétitive** suivant qu'une ou plusieurs valeurs sont possibles. On pourrait également parler de propriété **atomique ou décomposable**. Cela réfère au mécanisme d'agrégation déjà cité. Une propriété décomposable est un agrégat de propriétés atomiques et ou décomposables.

Pouvoir définir avec précision les **relations entre classes** constitue un atout de modélisation sémantique considérable. Cet atout peut toutefois rendre la phase d'intégration plus complexe lorsque les caractéristiques des propriétés ou relations entre classes sont différentes, voire contradictoires. Nous distinguerons ce **critère 6** du précédent du fait que certains modèles ne font pas la distinction entre classes et relations entre classes.

Critère 7: Langage de définition et de manipulation de données

On peut donner deux caractéristiques fondamentales: la **complétude** du langage et son **indépendance** vis-à-vis des aspects techniques de la BD.
Par complétude, on entend que la sémantique du langage soit assez riche pour exprimer les requêtes de tous les niveaux externes. En effet, ce langage devra permettre la consultation de chaque pièce d'information stockée dans la BD. Etant donné qu'à tous les niveaux de l'architecture on travaille sur des schémas conceptuels, c'est-à-dire définis indépendamment des caractéristiques de la BD, le langage devra faire abstraction de ces caractéristiques.
On distingue deux types de langages très utilisés, ceux de type ***procédural*** et ceux de type ***assertionnel***. Les langages procéduraux (ex: Codasyl DML) ne conviendront pas pour notre problème pour les raisons suivantes:

1. Il est difficile et long de poser une question adéquate dans un langage de programmation sans disposer d'une certaine compétence dans ce domaine. La technique la plus utilisée en programmation est l'itération. Celle-ci nécessite la définition d'un critère de terminaison. Si ce dernier n'est pas correct, le résultat peut être incomplet, reprendre du bruit(données incorrectes) ou ne jamais être fourni(silence) si le critère n'est jamais vérifié.(Boucle infinie)

2. Il est plus difficile de raisonner sur les langages de type procédural.

3. Il est également plus difficile de transformer des requêtes définies dans un langage de ce type. Or, nous verrons dans les chapitre suivants(4.3,4.5) que l'on sera confronté au problème de transformation de requêtes.

4. Les programme de ce type ne sont pas optimisables.

5. Ce type de langage est du type navigationnel et ne traite donc qu'un objet à la fois.

6. Ces langages sont proches des caractéristiques de la BD.

Par contre, les langages assertionnels sont plus susceptibles de répondre à nos exigences. En effet, les requêtes sont plus faciles à poser étant donné qu'elles le sont dans un langage spécifique, limité en commandes. De plus, ce type de langage est sujet à des algorithmes d'optimisation, travaille sur plusieurs objets à la fois et fournit toujours un résultat.
Parmi les langages assertionnels, on trouve des langages de type ***algébrique*** et de type ***calcul***. Ceux de type **algébrique** sont constitués d'un ensemble d'opérateurs permettant de définir explicitement la ***construction*** des résultats souhaités. Ce langage se caractérise donc par sa capacité à permettre de définir comment on désire construire le résultat. On spécifie le "comment construire", la procédure de résolution. C'est une formulation dite prescriptive.
Par contre, les langages de type **calcul** fournissent une notation permettant de définir le ***résultat*** souhaité en terme de données existantes. On définit le résultat sans donner la procédure d'aboutissement à ce résultat. On spécifie donc le "quoi construire". C'est une formulation dite "descriptive".
Nous accorderons notre préférence aux langages de type calcul qui rendent la tâche de l'utilisateur plus facile et qui permettent d'optimiser le traitement des requêtes. Ainsi, **quatre opérateurs** doivent être offerts par le langage: opérateur d'extraction(E) de données, d'insertion(I) de données, de modification(M) de données, de suppression(S) de données. Le format de ces requêtes serait du type SQL, c'est-à-dire composée de trois parties: l'*opérateur*(E,I,M,S), les *objets* sur lesquels portera l'opérateur et la condition de sélection des objets. Il s'agit d'une formulation générale pouvant souffrir des exceptions suivant les opérateurs.
Il devra être possible de définir des conditions de sélection constituées d'***opérateurs de comparaison***(<,>,≥,≤, =, ><) et d'***opérateurs logiques***(et, ou). La condition de sélection pourra

également être composée d'une requête, afin de définir des ***requêtes imbriquées***.

4.2.4. Critique des modèles existants

Il nous sera impossible de faire une critique complète de tous les modèles existants à l'heure actuelle étant donné leur nombre élevé. On se contentera d'exposer les paradigmes de modélisation et d'y reprendre les modèles qui sont les plus connus et donc les plus rencontrés dans la littérature.
On retrouve **cinq paradigmes** de modèle: hiérarchique, réseau, relationnel, entité-association, orienté-objets.

Le tableau 4.1 nous fournit une grille comparative des modèles sur bases des critères 1 à 7 exposé à la section 4.2.3.
Chaque ligne du tableau correspond à un modèle particulier. Chaque colonne du tableau reprend une des caractéristiques générales ou spécifiques.
L'intitulé complet des colonnes est le suivant:

Colonne 1 : Présence du mécanisme d'Instanciation - Classification

Colonne 2 : Présence du mécanisme d'Agrégation cartésienne (Cart.) et de couverture(Couv.)

Colonne3 : Présence du mécanisme de Spécialisation- généralisation (G), de propriétés (Prop.) telles que couverture, disjonction, partition et de l'héritage descendant(HD) et ascendant (HA)

Colonne 4 : Diversité des domaines: présence du domaine de base(Base) et des domaines construits(Const.)

Colonne 5 : Caractéristiques spécifiques aux propriétés des classes:
Atomique-Décomposable (AD)
Simple - Répétitif (SR)
Facultatif ou obligatoire (FO)

Colonne 6 : Possibilité de définir des contraintes d'intégrité pour affiner la sémantique

Colonne 7 : Caractéristiques des LMD et LDD: navigationnel(N), assertionnel de type algébrique(AA), assertionnel de type calcul(AC).

Colonne 8 : Possibilité de définir des schémas externes comme des sous-schémas (SS) du schéma global ou autres que des sous-schémas(Autr.)

Le tableau récapitulatif reprend dans sa dernière colonne quelques renseignements concernant la possibilité de définir des schémas externes dont l'intérêt a été soulevé lors de la définition des fonctions d'un modèle.

Ce tableau comparatif(tableau 4.1) nous fournit de précieuses indications: les modèles réseaux et hiérarchiques sont à bannir en tant que modèle canonique.
Analysons les autres modèles de plus près. Notons tout d'abord que les **modèles relationnels** ont largement été utilisés comme modèle canonique. Les raisons à cela étaient sans aucun doute la grande popularité de ce modèle, sa simplicité - origine à sa familiarité -, sa base théorique complète (algèbre et calcul relationnel, normalisation de relations). Cette dernière a permis de développer des langages de manipulation simples et performants. L'inconvénient de ce type de modèle est sa faible puissance de structuration. Peu de mécanismes d'abstraction y sont disponibles. Cela rend en effet difficile la modélisation d'objets complexes telles qu'on les rencontre aujourd'hui avec l'évolution de l'informatique vers les *Computer Aided Design (CAD)*, les *Computer Aided Manufacturing (CAM)*, les *Computer Integrated Manufacturing (CIM)*, et toutes les disciplines *"assistées par ordinateur"*(PAO,..) et les systèmes experts. Le modèle a connu des extensions (RM/T) mais le paradigme relationnel reste néanmoins inadéquat pour modéliser les objets complexes et donc comme base à l'intégration. L'absence de contraintes d'intégrité renforce également notre sentiment à l'égard du modèle relationnel. De plus le modèle relationnel reste néanmoins trop proche de l'implantation physique(concept de table); or nous avons dit notre souhait d'avoir un modèle conceptuel, c'est-à-dire plus abstrait. Cette critique n'avait pas pour but de savoir si ce modèle était bon ou mauvais en général mais bien de savoir s'il répondait aux exigences d'un environnement de BDH. A cette question, nous avons répondu négativement.

Les **modèles de type entité-association** et leurs extensions offrent quant à eux de nombreux mécanismes d'abstraction (spécialisation, généralisation, agrégation, ...). Ils sont également appréciés du fait de leurs représentations graphiques facilement compréhensibles. Cependant la dichotomie de représentation, type d'entités versus type d'associations, fait apparaître des problèmes au sein de la traduction et de l'intégration de schémas. Il faudra souvent choisir entre l'un des deux concepts de base et cela peut être un choix délicat. De plus l'absence de théorie, comme celle à la base du modèle relationnel, et l'inexistence d'une algèbre entité-association complète sont des lacunes qui sont encore à combler. Cependant certaines études ont fait état d'interprétation relationnelle des concepts des modèles entité-association [HAINAUT 90] pour fournir une base théorique à ces modèles. D'autres portent sur la définition d'une algèbre entité-association [PARENT 89] mais elle est limitée aux types d'entités. Il est en effet complexe de définir une algèbre dans un contexte présentant deux concepts de base.
Par contre, ce type de modèle permet de définir des contraintes d'intégrité(statiques et dynamiques) de divers types afin d'affiner la sémantique du schéma. On connaît en effet les contraintes sur les attributs(identifiant,...), les contraintes(d'inclusion ou d'exclusion) sur les types d'associations, les contraintes sur les rôles d'un type d'entité,...La panoplie est assez longue et très utilisée dans la pratique.

Tableau 4.1. : Tableau récapitulatif des critères de comparaison pour différents modèles

Caractéristiques	Inst°- Classif°	Agrég°- Décom°		Spéc°-Généralisat°				Domaines		Propriétés			Contr. Intég.	LMD/LDD		Sch. Ext.	
		Cart.	Couv.	G	Prop.	HD	HA	Base	Const.	AD	SR	FO		N	AC AA	SS	Autre
MODELES																	
Hiérarchique	oui	non	non	non	———			oui	non				(oui)	N		non	non
Réseau(Codasyl)	oui	oui	non	non	———			oui	non	oui	oui	non	(oui)	N		oui	non
Relationnel base	oui	non	non	non	———			oui	non	non	non	non	(oui)	AC	AA	oui	oui
Relationnel RM/T	oui	oui	oui	oui	non	oui	non	oui		non	oui	non	(oui)	AC	AA	oui	oui
Ent-Assoc de base	oui	non	non	non	———			oui	non	non	non	non	oui	———		non	non
Ent-Assoc Etendu	oui	oui	oui	oui	non	oui	non	oui	oui	oui	oui	oui	oui	AC		non	non
Ent-Assoc-Catég.	oui	oui	oui	oui	oui	oui	non	oui	oui	oui	oui	oui	oui	AA		non	non
ERC +	oui	oui	oui	oui	non	oui	oui	oui	oui	oui	oui	oui	oui	AA		oui	non
Fonctionnel(Daplex)	oui	oui	oui	oui	non	oui	non	oui	non	non	oui	non	non	AA		oui	non
Orienté-objet	oui	oui	oui	oui	oui	oui	non	oui	oui	non	oui	non	non	N		non	non
Frame- Slots	oui	non	non	oui	non	oui	non	oui	oui	non	non	non	non	AC		non	non

PS: Un "(oui)" signifie que la propriété traitée est faiblement représentée. Rappelons que ce tableau ne se veut pas exhaustif. On aurait pu analyser bon nombre d'autres modèles. On s'est contenté d'analyser les grands paradigmes de modélisation.

Du point de vue des langages de manipulation, les qualités de représentation graphique du modèle pourrait nous conduire à préférer une interface de manipulation de type graphique plutôt qu'un langage de type calcul.
Les modèles entité-association ont acquis une grande expérience depuis plus de 15 ans et sont très utilisés. Il suffit de voir la quantité de littérature y faisant référence aujourd'hui.

Les **modèles fonctionnels** offrent des mécanismes d'abstractions très intéressants, tels que la généralisation et l'agrégation mais restent encore limités au niveau des caractéristiques des propriétés des classes et des contraintes d'intégrité. De plus ce type de modèle offre une représentation graphique peu agréable et d'après la littérature a été peu convoité par les concepteurs de schémas.

Les modèles **orienté-objet** (OO) semblent également répondre aux exigences du modèle canonique. Tout d'abord leurs unicité de structuration, au travers du concept d'objet (atomique ou composant) et leurs mécanisme de composition permet de définir des objets complexes. A cela, on peut ajouter l'agrégation de couverture, construction typique en orienté-objet, qui permet également de définir des objets complexes. De plus parce que la base de ce modèle n'est plus la théorie ensembliste, comme pour les deux catégories de modèles précédentes, d'autres mécanismes d'abstraction, tel l'*amas* de données, y sont disponibles. Les mécanismes de classification permettent de définir des types de données abstraits, ceux d'héritage entre classes des schémas plus simples et ceux de composition d'objets des objets complexes. De plus, la technique d'encapsulation - technique qui consiste à regrouper la définition de la structure et celle du comportement - propre à l'orienté-objet est un avantage considérable. Il permet en effet de définir des opérations (*méthodes*) propres à chaque objet. Par exemple: l'objet CLIENT avec comme méthodes CREER-CLIENT, PASSER-COMMANDE... L'absence de théorie dans le cadre des modèles orienté-objets est toutefois un problème important; cela est dû à la récente naissance de cette nouvelle catégorie de modèle. Les modèles orienté-objet sont toutefois sémantiquement moins puissants que les modèles entité-association. En effet, la notion fondamentale d'identifiant et de contraintes d'intégrité ne s'y trouve pas.
La représentation graphique qu'offre les modèle OO est encore primitive et ne vaut pas celle des modèle EA.
Du point de vue des langages de manipulation, il faut avouer qu'il n'y a pas de langage algébrique ou calcul disponible aujourd'hui. Cela est essentiellement dû à la technique d'encapsulation propre à l'orienté-objet. Cette dernière limite les opérations autorisées sur les objets en définissant des méthodes propres à chaque objet. Il n'existe donc pas de langage de manipulation au sens défini antérieurement mais bien un langage de programmation permettant d'utiliser les méthodes des objets. Ce langage est de type navigationnel et donc ne convient pas pour notre problème.
Nous espérons donc que la recherche continuera à développer ce modèle afin d'accroître sa puissance de modélisation et de développer une algèbre associée. [ROLLAND 92] propose un modèle O* reprenant les concepts de contraintes d'intégrité. Les modèles OO seront probablement d'excellent représentant au poste de modèle canonique dans les années à venir.

Afin d'être complet, nous finirons sur le modèle **FRAME/SLOT** qui a fait l'objet d'une brève

présentation dans l'architecture de CARNOT(section 2.3. B). Ce type de modèle offre peu de mécanismes d'abstraction, de propriétés d'attributs et de contraintes d'intégrité. Par contre, il offre de puissants mécanismes d'inférence et un langage basé sur le calcul des prédicats du premier ordre. Ce modèle ne semble donc pas adapté au problème qui nous concerne. Ce type de modèle est reconnu comme plus adapté aux problèmes de représentations de connaissances, très rencontrés dans le domaine de l'intelligence artificielle.
Nous pouvons conclure de cette analyse que les modèles hiérarchique, réseau et relationnel ne conviennent pas pour remplir les fonctions d'un modèle canonique. Les modèles orienté-objet conviendraient parfaitement s'ils étaient complétés de telle sorte à les rendre sémantiquement plus riches et à permettre la définition de langage de manipulation assertionnel. Toutefois, cela faisant encore défaut, on optera pour un modèle de type entité-association accompagné d'extensions de modélisation. Ce dernier présente en effet de puissants mécanismes de représentation de données et s'avère apte à faire face à de nombreux modèles de SBD locaux. Ce modèle sera présenté à la section suivante.
Rappelons que le modèle canonique permet de résoudre le problème d'*hétérogénéité syntaxique*(cfr section 1.2).

4.2.5. Exposé du modèle canonique

Le modèle que nous avons choisi est donc un modèle de type **entité-association étendu** par des variantes de modélisation. Nous l'appelerons modèle EA/E.
Nous avons repris des concepts et représentations graphiques dans différents modèles EA et notamment dans [PARENT 87], [HAINAUT 91a].

A. Entité et type d'entité(TE)

La notion d'entité est fondamentale. Elle est généralement définie comme tout concept concret ou abstrait de l'univers de discours(personne, objet,...) que l'on reconnaît comme individualisable et digne d'intérêt en lui-même ou plus concrètement comme la représentation d'un objet de l'univers de discours. Toute entité appartient à une ou plusieurs classes, appelées types d'entités. Les classes sont définies de telles sorte que les entités qu'elles recouvrent jouissent des propriétés structurelles(attributs et rôles) identiques. Un type d'entité porte un nom qui l'identifie parmi tous les TE du réel perçu.

B. Association et type d'association(TA)

Les entités sont en association les unes avec les autres. Une association est définie comme un groupe de deux ou plusieurs entités, chacune d'elles jouant un rôle défini dans ce groupe. Une association est un ensemble d'au moins deux rôles, chacun d'eux étant joué par une et une seule entité(une même entité peut d'ailleurs jouer plus d'un rôle dans une même association). Toute association appartient à une et une seule classe, nommée type d'association. Les classes sont définies de telle sorte que les associations qu'elles recouvrent jouissent des propriétés structurelles (attributs et rôles) identiques. Le degré d'un type d'association est le nombre de rôle de chaque association de ce type. Un type d'association a un nom qui l'identifie parmi tous les

TA de l'univers de discours.

C. Rôle d'un type d'association

Toutes les associations d'un type ayant le même rôle, on peut parler des rôles d'un TA. Toutes les entités jouant un rôle déterminé dans les associations d'un type appartiennent au même TE. Un rôle porte un nom qui l'identifie parmi tous les rôles que joue le type d'entité. Afin de préciser le nombre d'associations dans lesquelles une entité peut apparaître, on attache à chaque rôle une **contrainte de connectivité** (appelé parfois **cardinalité**). Celle-ci s'exprime sous la forme de deux entiers i-j qui spécifient que toute entité pouvant jouer ce rôle devra le jouer un nombre de fois compris entre i et j. Une valeur N attribuée à j représente une valeur infinie. Graphiquement (Section F ci-dessous), on ne représentera que les contraintes les plus courantes, à savoir 0-1,1-1,0-N,1-N,M-N. On pourra toutefois préciser les valeurs de M et N lorsque cela s'avère nécessaire.

D. Domaines de valeur

Un domaine ou type de valeurs est un ensemble invariable de valeurs de même structure et obéissant aux même lois de manipulation.
On considérera des types de bases admis par tous: entier, réel, chaine de caractères, booléen mais également des types construits tels que le type énuméré ou liste: JOUR ("Lundi", Mardi", "Mercredi", "Jeudi", Vendredi", "Samedi", "Dimanche"). L'analyse des attributs nous indiquera encore d'autres domaines définis par construction ou composition.

E. Les attributs

Un attribut d'un TE est constitué de l'association d'un domaine de valeur à ce type d'entités. Il permet de représenter par une ou plusieurs valeurs une propriété caractérisant chaque entité de ce type.
Un attribut d'un TA est constitué de l'association d'un domaine de valeur à ce TA. Il permet de représenter par une ou plusieurs valeurs une propriété caractérisant chaque association de ce type.
Un attribut pourra être **obligatoire** ou **facultatif** suivant que toute entité ou association doit ou pas être dotée d'une valeur au moins de l'attribut. Il sera **décomposable** ou **atomique** selon qu'il est possible ou non de décomposer chaque valeur en fragments significatifs. Il est **simple** ou **répétitif** suivant qu'on admette au plus une valeur par entité ou plusieurs valeurs par entité.

F. Représentation graphique des composants présentés

La figure 4.4. représente donc les

TE CLIENT, COMMANDE, PRODUIT
TA COMMANDE-CLIENT, LIGNE-COMMANDE
Rôles "passe" de connectivié 0-N, "passée-par" 1-1, "porte-sur"1-N,

"référencé-par" 0-N
attribut facultatif répétitif "Téléphone"
attribut répétitif obligatoire "Prénom"
attribut décomposable "Adresse", les autres attributs sont atomiques, simples et obligatoires

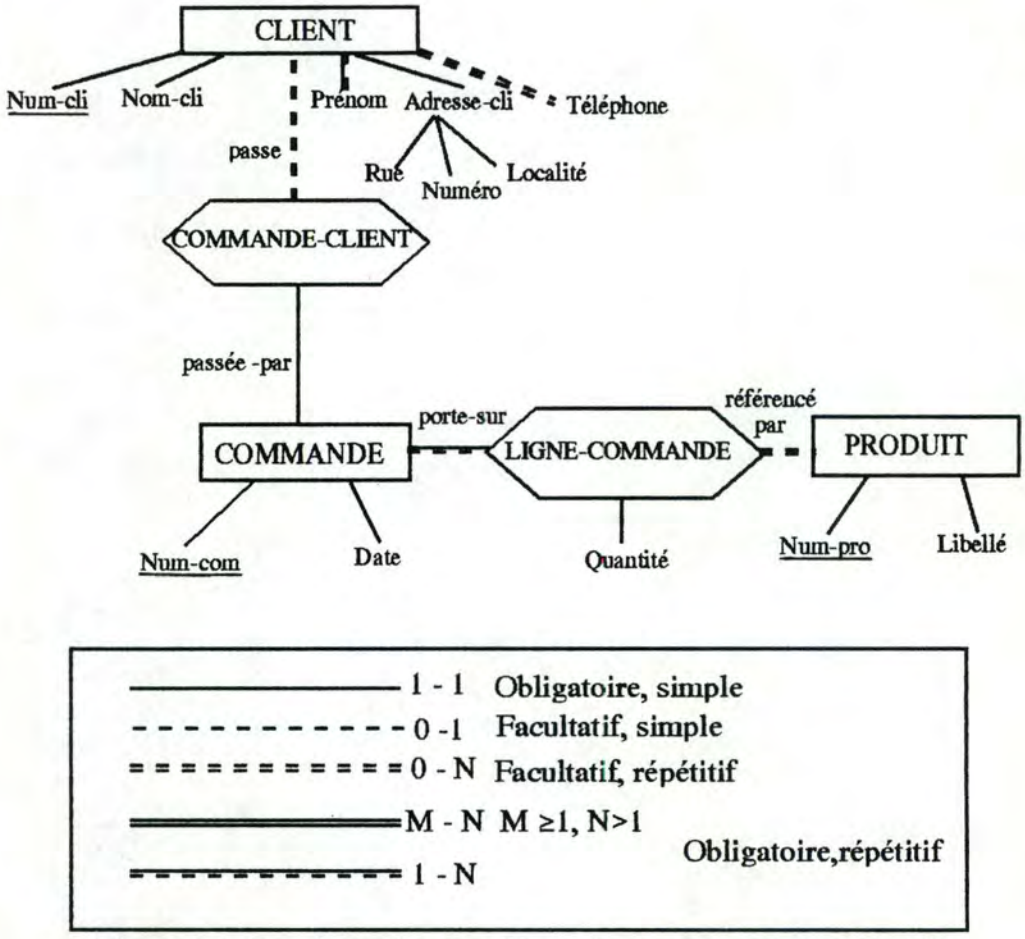


Figure 4.4. Représentation graphique des concepts de bases EA/E

G .Identifiants et contraintes d'intégrité

Un **identifiant d'un TE** E est un groupe constitué d'attributs de E et/ou de rôles que jouent d'autres TE dans les types d'associations dans lesquelles E apparaît dans un rôle déterminé. A une *valeur* d'un identifiant, correspond au plus une entité. Une valeur d'un identifiant est constituée d'une valeur pour chaque constituant attribut et une entité pour chaque constituant rôle.

Un **identifiant d'un TA** A est un ensemble de rôles et/ou d'attributs de A tels qu'à tout instant il n'existe pas plus d'une association contenant les mêmes entités et les mêmes valeurs pour ces constituants.

Dans la mesure du possible, on soulignera l'identifiant des TE et TA par un trait continu. Si cela s'avère impossible, on le notera explicitement sous la forme: "Id(TE)= suite d'attributs et/ou de

rôles"

Font également partie des contraintes d'intégrité, les **contraintes de cardinalité des attributs** et les **contraintes de connectivité des rôles des TE.**

Il existe d'autres types de contraintes, telles que les contraintes sur les attributs, les types d'associations et les rôles.
Il s'agit de contraintes portant sur des relations logique entre attributs(=, <,>, <>, ≥, ≤), entre rôles joués par un même TE (inclusion, exclusion, égalité de rôles)ou entre types d'associations de mêmes TE.(inclusion, égalité, exclusion de TA)

H. Concepts d'extension

H.1. Généralisation-spécialisation

De types d'entités

Ce mécanisme permet définir des relations d'inclusions entre populations de classes distinctes. Sur la figure 4.5 la population du TE générique VEHICULE inclu l'ensemble des populations des TE spécifiques VOITURE, CAMION et MOTO. On appelle également "lien is-a" cette relation de généralisation-spécialisation entre ce TE générique et ces TE spécifiques; en effet, une VOITURE is-a (est-un) VEHICULE, un CAMION is-a VEHICULE,...
Associé à ce mécanisme, on trouve l'héritage descendant (du type générique vers le type spécifique). La figure 4.5. illustre ce mécanisme.

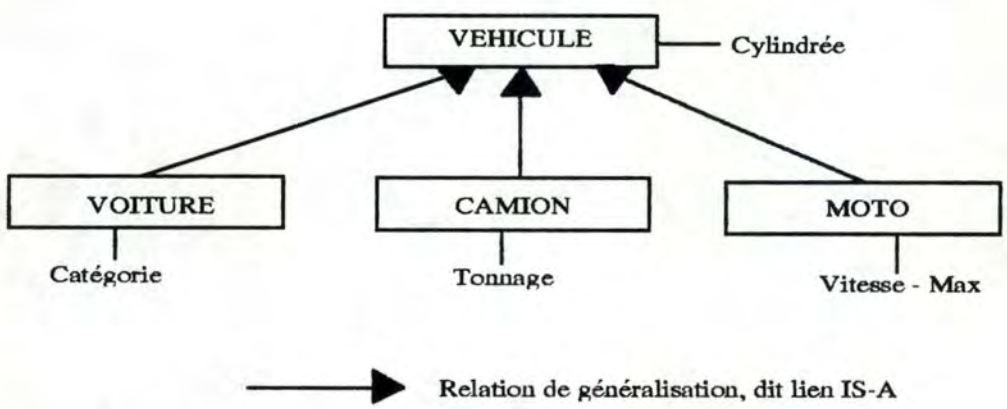


Figure 4.5. Représentation des relations de généralisation entre TE

Ainsi, on verra comme véhicule toute voiture, camion ou moto. Ces trois TE hériteront de l'attribut "cylindrée", relatif au TE véhicule. Ainsi, on pourra parler le la cylindrée d'une voiture, d'une moto ou d'un camion. Par contre, les attributs "catégorie", "tonnage" et "vitesse-max" sont propres aux TE auxquels ils sont rattachés.

Rappelons la propriété de **transitivité** de la relation IS-A: si Assistant IS-A Etudiant et Etudiant IS-A Personne alors par transitivité Assitant IS-A Personne.

Nous offrons la **généralisation multiple** (à un type d'entité spécifique peut correspondre plus d'un type d'entité générique) ainsi que trois propriétés sur les généralisations. Il s'agit des propriétés de couverture, de disjonction et de partition. Nous n'offrons pas de représentation graphique car celle-ci risquerait de rendre le schéma illisible. Elles seront exprimées à l'aide de contraintes d'intégrité en utilisant le notations ensemblistes, comme suit:

Couverture: "Tout véhicule est une voiture et/ou un camion et/ou une moto"

Véhicule = Voiture ∪ Camion ∪ Moto

Disjonction: " Une voiture n'est pas un camion"

Camion ∩ Voiture = ø

Partition : "Tout véhicule est une voiture ou un camion ou une moto"

Véhicule = Voiture ∪ Camion ∪ Moto

Voiture ∩ Camion ∩ Moto = ø

De types d'associations

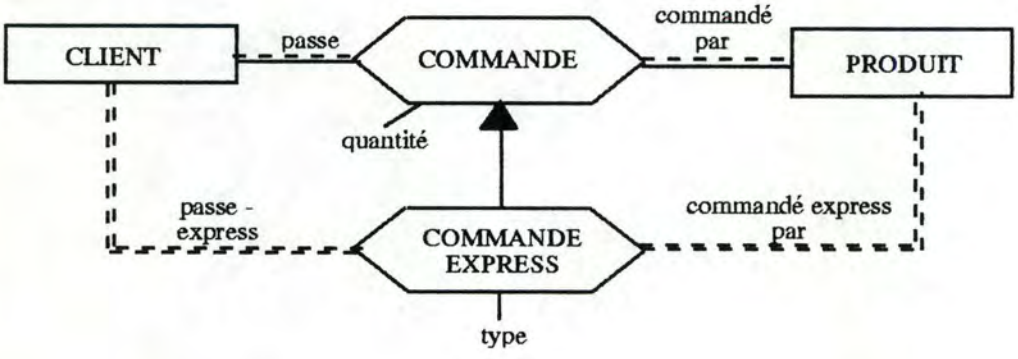


Figure 4.6. Représentation des relations de généralisation entre TA

La signification de ce mécanisme est identique à celui concernant les TE, excepté que l'héritage porte à la fois sur les attributs et les rôles du(des) TA génériques .

H.2. Agrégation cartésienne et de couverture

Le mécanisme d'agrégation **cartésienne** a déjà été exposée par l'intermédiaire des attributs

décomposables. Ce mécanisme permet de définir des objets complexes; on parle alors de hiérarchie d'agrégations (L'attribut "Adresse" de la Figure 4.4). Le type d'association est un exemple d'agrégat cartésien (attributs et rôles).

Insistons sur le mécanisme d'agrégation de **couverture**. Ce dernier permet de regrouper sous une entité (agrégat) un ensemble d'entités de types différents(composants).
La figure 4.7 illustre ce concept d'agrégation de couverture et se lit comme suit: une entité flotte est constituée de 1-N entités Navires et de 1-N entités Militaire et de 0-N entités Avion. Pour exprimer la cardinalité des TE composants, on utilise la même représentation graphique que celle utilisée pour les rôles et les attributs. En revanche, une entité navire, militaire ou avion ne doit pas nécessairement faire partie d'une flotte. Cette relation est implicite pour tout composant d'un agrégat de couverture.
Aucun mécanisme d'héritage n'est associé à cette structure d'agrégation.

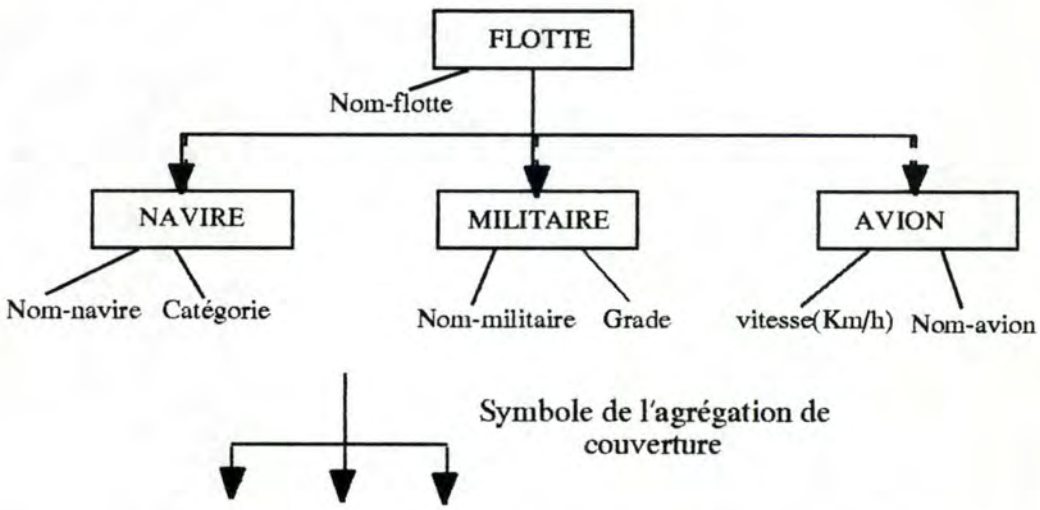


Figure 4.7. : Représentation de l'agrégation de couverture

H.3. Rôles multi-domaines

Ce mécanisme permet d'alléger les schémas et de permettre à un rôle d'être joué par plusieurs types d'entités.

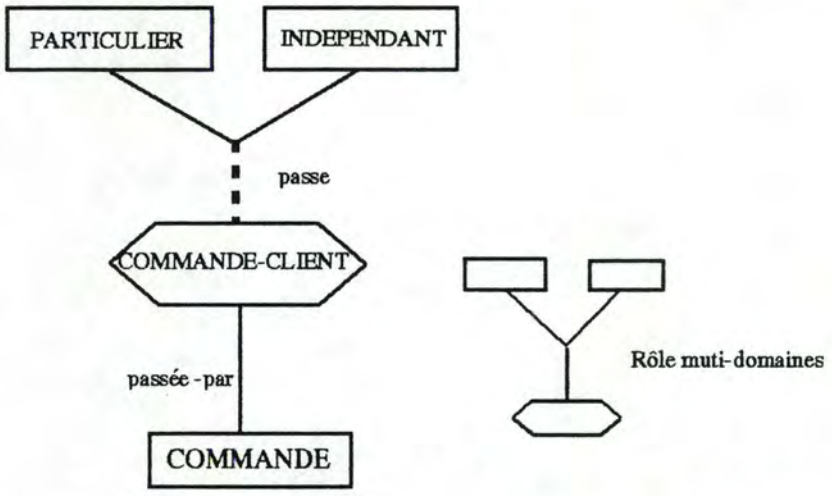


Figure 4.8. Représentation de la construction "rôle multi-domaines"

Ainsi, une commande peut être passée par un particulier **ou** un indépendant. Un indépendant ou un particulier peut passer 0 à N commande(s) dans le cas présent.

D'autres mécanismes de modélisations sont disponibles dans la littérature. On s'est restreint à ce sous-ensemble pour ne pas surcharger les schémas de constructions différentes et pour ne pas trop complexifier les tâches ultérieures telles que l'intégration. On aurait tendance à dire que les deux dernières constructions(H.2 et H.3) doivent être réservées pour résoudre des problèmes d'intégration exclusivement.

I. Les langages de définition et de manipulation

Par référence au modèle EA/E, nous regrouperons ces langages sous l'appelation **L-EA/E.**

I.1. Langage de définition

Pour désigner l'attribut composant a2 de l'attribut décomposable $A(a_1,a_2,..,a_n)$, on fera comme suit: $A.a_2$.

| : ou exclusif
\ : l'un ou l'autre ou les deux
Symbole en gras : symbole obligatoire comme "**CREATE**" ou "**{}**" ou "**,**" ou "**:**"
expression en italique: à remplacer par un des éléments repris dans sa définition
$[X]^{i,j}$: l'expression *X* doit figurer au minimum i fois et au maximum j fois. On séparera les occurences par des virgules. L'absence d'indices i et j signifie que l'expression *X* est facultative (i=0, j=1)
a,b: couple de nombres entiers tels que b ≥ a
i,j: couple de nombres réels tels que j≥ i
max: nombre entier
relation logique : | < | > | <>| ≤ | ≥| =
A,B,...,Z, nom-TE, nom-TA, nom-TE spécifique, nom-TE-générique, nom-TE-agrégat, nom-TE-

composant, nom-attribut, nom-rôle, nom-TA-spécifique, nom-TA-générique: chaînes de caractères

expr-attribut : *nom-attribut* {*a,b*} : *domaine*

expr-rôle : [*nom-TE*]^{1,N} : *nom-rôle* {*a,b*}

domaine = **INTEGER** [{*i..j*}] | **REAL** [{*i..j*}] | **STRING** [{*max*}] | **BOOLEAN** | {A, B, C,..Z} | **AGREGAT** { [*expr-attribut*]^{1,N} }

CREATE ENTITY-TYPE *nom-TE*
[**WITH** [*expr-attribut*]^{1,N}]

CREATE RELATIONSHIP-TYPE *nom-TA*
[**WITH** [*expr-attribut*]^{1,N}]
BETWEEN [*expr-rôle*]^{2,N}

CREATE SPECIALISATION-ET
BETWEEN [*nom-TE-spécifique*]^{1,N} **;** *nom-TE-générique*

CREATE SPECIALISATION-AT
BETWEEN [*nom-TA-spécifique*]^{1,N} **;** *nom-TA-générique*

CREATE COVER-AGREGAT
BETWEEN *nom-TE-agrégat*; [*nom-TE-composant* {*a,b*}]

DEFINE [*nom-attribut* \ *nom-rôle*]^{1,N}
AS IDENTIFIER OF *nom-TE* | *nom-TA*

DEFINE COVER
BETWEEN [*nom-TE-spécifique*] ^{1,N};*nom-TE générique*

DEFINE DISJONCTION | PARTITION
BETWEEN [*nom-TE-spécifique*] ^{2,N};*nom-TE générique*

I.2. Langage de manipulation

NB: Les opérations proposées constituent un noyau minimal du langage, qui pourrait être étendu à d'autres fonctions.

On utilisera les mêmes notations que pour la spécification du langage de définition(*I.1*)

valeur-attribut : valeur issue du domaine de l'attribut

expression-attribut : *valeur-attribut* | *fonction mathématique applicable au domaine de l'attribut (substring,addition,soustrcation, multiplication,...)*

condition de sélection:
nom-attribut relation-logique valeur attribut | *nom-attribut*
| *nom-attribut* **IN SELECT** *nom-attribut*
FROM [*nom-TE/nom-TA*]^{1,N}
WHERE *condition de sélection*
| *nom-TE-1 nom-rôle nom-TE-2*
| *nom-TE- composant* **IN** *nom-TE-agrégat*
| *condition de sélection* **AND** *condition de sélection*

| *condition de selection* **OR** *condition de sélection*

liste-d'attributs: réfère à une liste d'attributs non nécessairement associés au même TE ou TA. Si il peut y avoir ambiguïté dans les noms, on placera le nom du TE ou TA en suffixe de l'attribut, suivi d'un "."

Quatre opérations sont définies:

SELECT *liste-d'attributs*
FROM [*nom-TE* \ *nom-TA*]^{1,N}
WHERE *condition de sélection*

ADD *nom-TE* [*nom-attribut* := *valeur-attribut*] ^{0,N}

ADD *nom-TA* **BETWEEN** [*nom-TE*]^{2,N} [**WITH** [*nom-attribut* := *valeur-attribut*] ^{0,N}]
WHERE *condition de sélection*

DELETE *nom-TE* | *nom-TA*
WHERE *condition de sélection*

MODIFY *nom-TE* [*nom-attribut* := *expression-attribut*]^{0,N}
WHERE *condition de sélection*

MODIFY *nom-TA* [*nom-attribut* := *expression-attribut*] ^{0,N} [**ON** [*nom-rôle* : *nom-TE* **WITH** *condition de sélection*] ^{1,N}]
WHERE *condition de sélection*

Exemples de définition de schémas

Reprenons les exemples des figures précédentes(4.4 à 4.7)

Sur base de la figure 4.4.

```
CREATE ENTITY-TYPE Client
WITH   Num-cli {1,1} : INTEGER,
       Nom-Cli {1,1} : STRING {25},
       Prénom {1,3} : STRING {20},
       Adresse-Cli {1,1} : AGREGAT { Rue {1,1} : STRING {20},
                                     Numéro {1,1} : INTEGER,
                                     Localité {1,1} : STRING{30}},
       Téléphone {0,N}: STRING{9}

CREATE ENTITY-TYPE Commande
WITH  Num-com {1,1} : STRING{10},
      Date{1,1} : STRING{8}

CREATE ENTITY-TYPE Produit
WITH  Num-pro {1,1} : STRING{10} ,
      Libellé {1,1} : STRING{20}
```

CREATE RELATIONSHIP-TYPE Commande-client
BETWEEN Client : passe {0,N} , Commande : passée-par {1,1}

CREATE RELATIONSHIP-TYPE Ligne-commande
WITH quantité{1,1} :INTEGER
BETWEEN Commande : porte-sur{1,N} , Produit : référencé par {0,N}

N.B.On suppose pour les exemples suivants que les TE sont déjà créés.

Sur la figure 4.5.

CREATE SPECIALISATION -TE
BETWEEN Voiture, Camion, Moto; Véhicule

Sur la figure 4.6.

CREATE SPECIALISATION-TA
BETWEEN Commande-express; Commande

Sur la figure 4.7.

CREATE COVER-AGREGAT
BETWEEN Flotte; Navire{M,N}, Militaire{1,N}, Avion{0,N}

Sur la figure 4.8.

CREATE ASSOCIATION-TYPE Commande-client
BETWEEN Commande : passée-par {1,1}, Particulier, Indépendant : passe{0,N}

Exemple de manipulation

Sur la figure 4.4.

"Donner les nom, prénom, adresse des clients de la localité de "TOURNAI", ayant commandé des produits de libellé "CHOCOLAT"; donner les numéro de commande correspondants."

SELECT Nom-cli, Prénom, Adresse, Num-Com
FROM CLIENT,COMMANDE, PRODUIT
WHERE Localité= "TOURNAI" AND CLIENT passe COMMANDE
AND COMMANDE porte-sur PRODUIT
AND Libellé= "CHOCOLAT"

Sur la figure 4.4.

"Ajouter une ligne-commande entre la commande de Num-com ='CD2123 ' et le produit de

Num-pro = 'P87' " à concurrence d'une quantité de 210."

```
ADD LIGNE-COMMANDE
BETWEEN COMMANDE, CLIENT
WITH Quantité : =210
WHERE Num-com = "CD2123" AND Num-pro = "P87"
```

Sur la figure 4.4.

"Modifier, pour toutes les commandes dont la date est postérieure au 31/12/91, toutes les lignes de commandes du produit P65 vers le produit P71 en diminuant la quantité de 10%"

```
MODIFY LIGNE-COMMANDE quantité := quantité *0.9
ON porte-sur : PRODUIT WITH Num-pro = "P71"
WHERE Commande.Date > 31/12/91 AND COMMANDE porte-sur PRODUIT
                                AND Produit.Num-pro = "P65"
```

Sur la figure 4.5.

"Donner le tonnage et la cylindrée de tous les camions de tonnage compris entre 25 T et 30 T"

```
SELECT Tonnage, Cylindrée
FROM VEHICULE, CAMION
WHERE (tonnage < 30 T) and (tonnage > 25 T )
```

Sur la figure 4.7.

"Donner les noms des navires et les noms des militaires contenus dans la flotte de nom "X412"

```
SELECT nom-navire, nom-militaire
FROM FLOTTE, NAVIRE, MILITAIRE
WHERE (NAVIRE IN FLOTTE) AND (MILITAIRE IN Flotte)
                         AND (Nom-flotte = "X412")
```

4.2.6. Modèle canonique et méta-modèle

Nous n'avons envisagé que la résolution du problème par l'utilisation d'un modèle de données. On peut toutefois aborder le problème en utilisant un **méta-modèle**, c'est-à-dire un modèle permettant de représenter, par des méta-méta-types, l'ensemble des méta-types qu'offrent les modèles de données. Ce mécanisme s'étend naturellement aux schémas et aux données; de ce fait, on peut facilement comparer les schémas.

Le méta-modèle permet de représenter dans un formalisme unique l'ensemble des concepts ou méta-types proposés par divers modèles de données. De ce fait, l'équivalence entre concepts de schémas est vérifiée si et seulement si les méta-méta-types correspondants sont équivalents.

Cettte approche de résolution ne fait que connaître ses débuts. C'est pourquoi on se limite à signaler l'existence de cette approche. On peut consulter [BARSALOU 91] et [ROLAND 92] qui s'attachent respectivement à l'exposé des méta-modèle M(DM) et obj_méta et de leurs applications.

Le modèle multi-niveaux M(DM) permet de représenter dans la logique du second ordre les concepts proposés par différents modèles. Ainsi deux concepts sont équivalents si leurs interprétations M(DM), en réalité des formules de logique de second ordre, sont équivalentes. L'approche multi-niveaux de obj_méta se base sur l'approche orienté-objet et essaie de ramener les méta-types des modèles de données à des "concepts" obj-méta.
la figure 4.9. illustre l'architecture multi-niveaux d'un système d'information mettant en évidence les liens entre instances, schémas, modèle et méta-modèle.

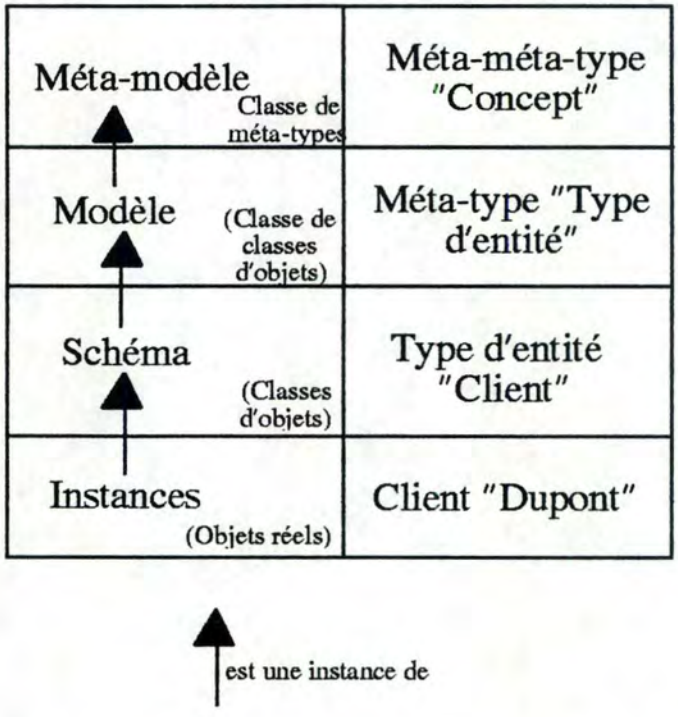


Figure 4.9: Architecture multi-niveaux d'un système d'information

4.3. Les problèmes de transformation

4.3.1. Introduction

Face à l'hétérogénéité syntaxique existant entre les SBD locaux, nous avons décidé de répondre par une phase d'homogénéisation des schémas locaux. Cette dernière consiste donc en la **transformation des schémas** locaux dans le formalisme du modèle canonique, à savoir celui de la section 4.2.5.
Aux modèles des SBD locaux sont associés des langages (LMD et LDD) spécifiques (SQL, CODASYL DML,...). Etant donné que les requêtes parviendront via les schémas composants[1], il faudra **transformer ces requêtes** dans le formalisme des langages des SBD locaux mais également en fonction des concepts des schémas locaux. Finalement, on constate que chaque modèle présente un format de données spécifique (table relationnel, record CODASYL,...) au moyen duquel il transmet ses données. Nous aurons donc à résoudre le problème de **transformations de données** dans le format du modèle canonique et en fonction des concepts proposés dans les schéma composants.
On a ainsi identifié trois processus de transformation. Remarquons l'étroite relation qui existe entre eux. En fait, la transformation des schémas est à l'origine des autres transformations. En effet, pour pouvoir transformer les requêtes et les données, il faut que l'on dispose d'information établissant la correspondance entre les schémas. Cette dernière ne peut être établie que par le processus de transformation de schémas et est regroupée en **règles de mapping**. Celles-ci définissent les concepts du schéma final en termes de concepts du schéma initial. On devra donc s'interroger sur la définition d'un formalisme de description des règles de mapping.
La figure 4.10 illustre les relations entre ces trois processus.

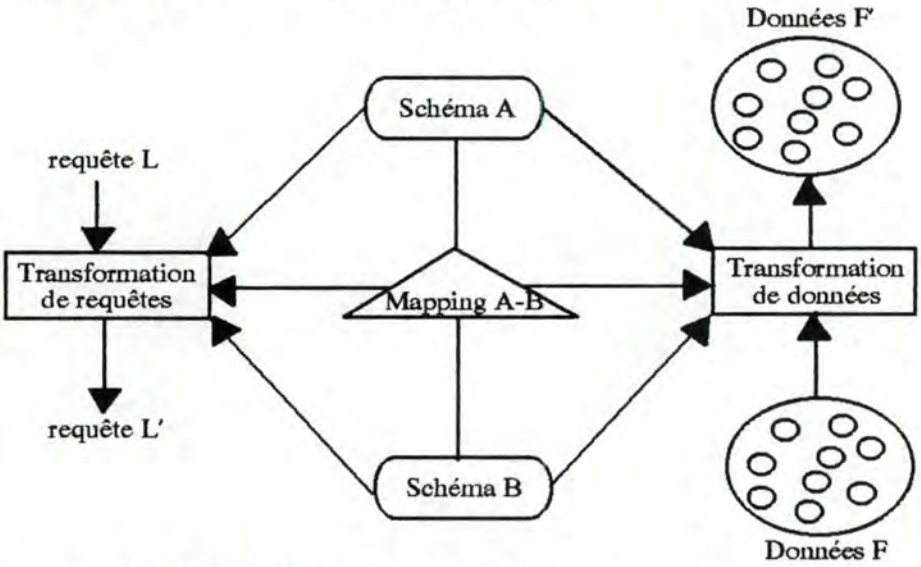


Figure 4.10 : Relations entre processus de transformation

Le processus de transformation de schémas se distinguent des autres processus de transformation par le fait qu'ils réalisent des opérations "**one-shot**" c'est-à-dire des opérations effectuées une fois pour toutes.

1. Le schéma composant correspond à la vue canonique(EA/E) d'un schéma local; Cfr. figure 3.1

Les autres opérations de transformations sont dites "**temps réel**" car elles doivent être réalisées à chaque nouvelle requête.

La transformation de schémas et de données est réalisée du bas vers le haut de la hiérarchie des schémas si l'on se réfère à la figure 3.1. du chapitre précédent. Par contre la transformation de requêtes se déroule du haut vers le bas de cette hiérarchie. Nous verrons dans la section suivante que cette dernière caractéristique va nous conduire à résoudre le problème de transformation de données en même temps que celui de transformations de schémas. Ce regroupement est naturel dans la mesure où les *données constituent les instances ou occurrences d'un schéma.*

Afin d'illustrer ces problèmes, reprenons l'exemple de la figure 4.1. En prenant comme modèle canonique, le modèle entité-association étendu (EA/E) de la section 4.2.5 et le langage L-EA/E, nous aurons à définir les processus de transformation suivant:

TS $_{A\text{-}B}$ = transformation d'un schéma de modèle A dans un schéma de modèle B

TR $_{L,A,M\ -\ L',B,M'}$ = transformation d'une requête en langage L sur un schéma A décrit dans le modèle M en une requête en langage L' sur un schéma B décrit dans le modèle M'

TS $_{REL\text{-}E/AE}$: schéma local [Relationnel] -> schéma composant [entité-association étendu]

TS $_{COD\text{-}E/AE}$: schéma local [Codasyl] -> schéma composant [entité-association étendu]

TS $_{Fonct\text{-}EA/E}$: schéma local[Fonctionnel] -> schéma composant [entité-association étendu]

TR $_{L\text{-}EA/E,\ Sch.\ EA/E,EA/E\ -\ SQL,\ BD1,\ relationnel}$: Requête [L-EA/E, Sch. EA/E, EA/E] ->
Requête [SQL, BD1, relationnel]

TR $_{L\text{-}EA/E,Sch.\ EA/E\ ,EA/E\ -\ Codasyl\ DML,\ BD2,\ Codasyl}$: Requête[L-EA/E, Sch.EA/E,EA/E] ->
Requête [Codasyl DML, BD2, Codasyl]

TR $_{L\text{-}EA/E,Sch.\ EA/E\ ,EA/E\ -\ DAPLEX,BD3,Fonctionnel\ daplex}$: Requête [L-EA/E, Sch.EA/E, EA,E] ->
Requête [Daplex, BD3, fonctionnel Daplex]

Ainsi, à partir des schémas de la figure 4.1., nous souhaiterions, intuitivement, obtenir après transformations les schémas de la figure 4.11. Nous détaillerons dans les sections suivantes comment on peut passer des schémas de la figure 4.1 à ceux de la figure 4.11.

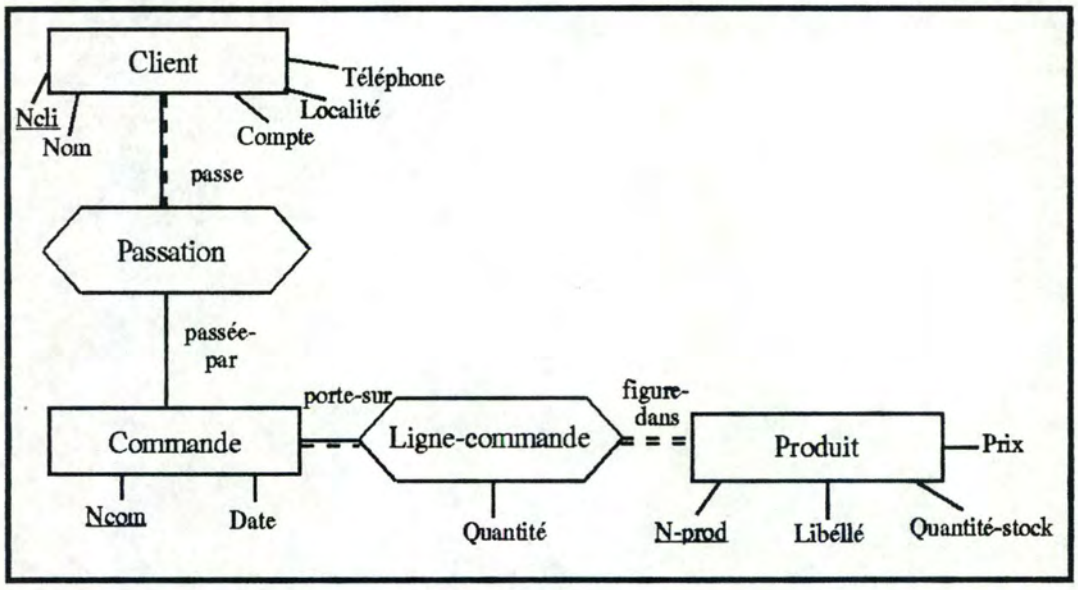


Schéma EA/E de la BD locale Relationnelle BD1

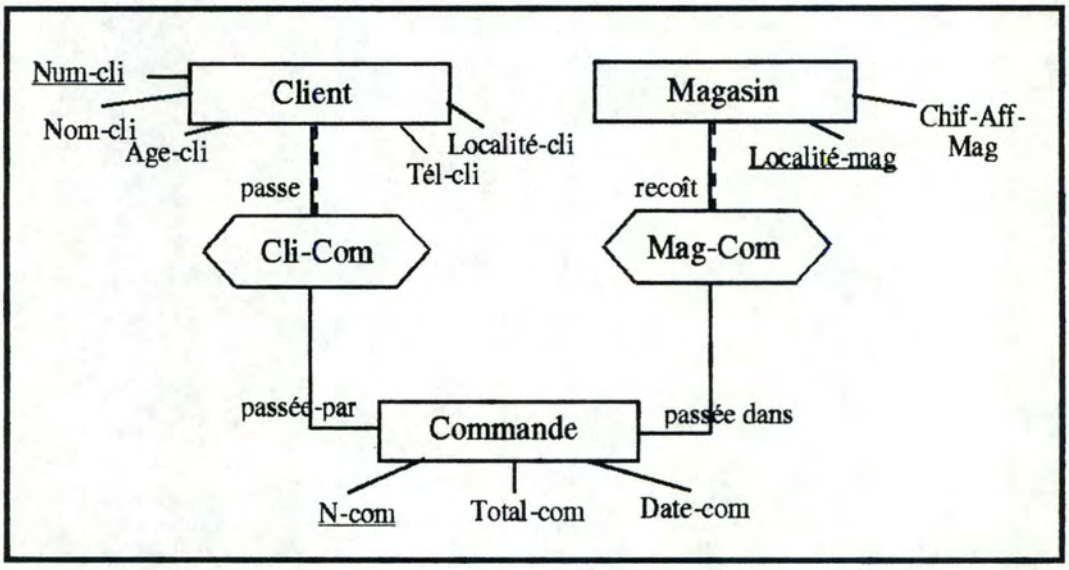


Schéma EA/E de la BD locale Réseau Codasyl BD2

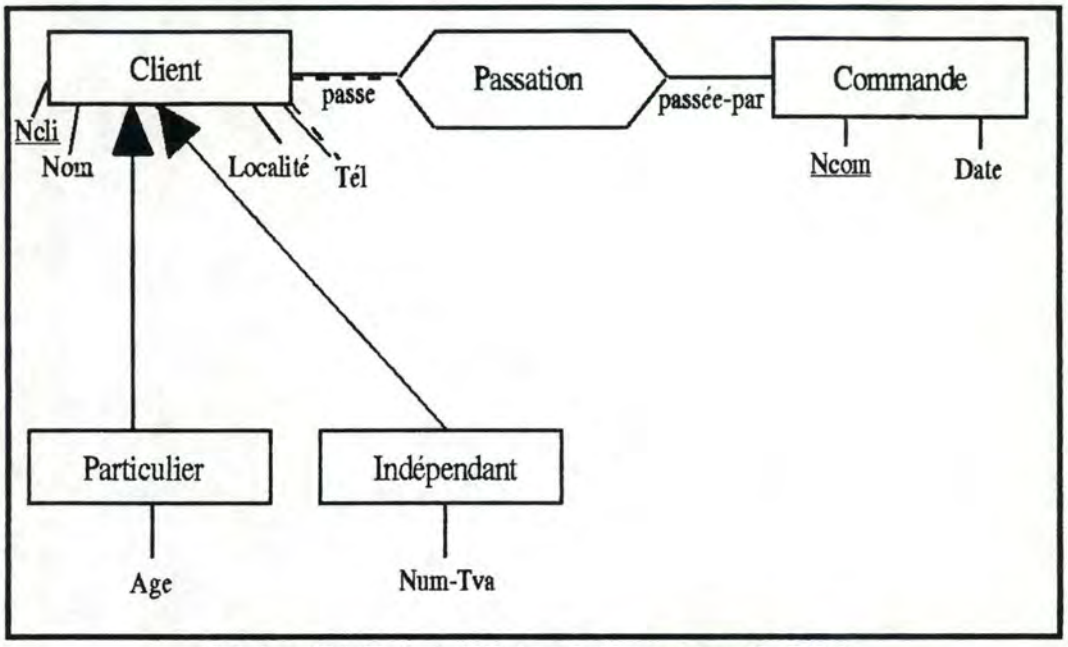


Schéma EA/E de la BD locale Fonctionnelle BD3

Figure 4.11.: Schémas EA/E des bases de données de la figure 4.1

Nous constatons que lors de l'élaboration de ces schémas EA/E, nous avons introduit une sémantique plus précise, notamment grâce aux contraintes de connectivité. Cela vaut particulièrement pour le schéma relationnel. On a créé un TA passation de rôles (1-N)(1-1) sur base de l'attribut N-cli de la relation Commande et de la contrainte référentielle associée. De même, on a créé un TA Ligne-commande de rôles (1-N)(0-N) pour la relation du même nom, alors que pour les deux autres relations nous avons préféré les TE. Nous nous sommes inspirés des contraintes référentielles entre les attributs des relations. On a ainsi précisé la sémantique du schéma initial. On dénomme généralement ce problème par l'expression "enrichissement sémantique".
Après avoir donné une définition formelle de la notion de transformation de schémas (4.3.2 A), nous exposerons la méthode de transformation (4.3.2 B) et ensuite nous spécifierons quelques règles de transformation pour le modèle relationnel et Codasyl en rapport avec le modèle EA/E (4.3.2 C).

4.3.2. Transformation de schémas et de données

A. Définitions

La transformation d'un schéma source S exprimé selon un modèle M est un processus qui produit un nouveau schéma, schéma cible, exprimé dans un formalisme M' (M=M' ou M <> M') et ayant une *sorte d'équivalence* avec le premier schéma.[HAINAUT 91b]

Par l'expression *"sorte d'équivalence"*, on entend une transformation qui préserve la sémantique du schéma initial c'est-à-dire que le schéma source et le schéma cible représentent exactement les mêmes faits de l'univers de discours mais dans des formalismes différents. Ces deux schémas sont donc **sémantiquement équivalents.**

Cette transformation est telle que toute instance du schéma source, entendons un ensemble de données, peut être convertie en une instance du schéma cible sans introduire de perte ni de bruit et inversément si la transformation est **réversible.**
Dans notre contexte, il est souhaitable que la transformation soit réversible afin que l'on puisse à la fois exécuter des opérations de consultation et de modification.
Donnons à présent une définition plus formelle de la notion de transformation réversible de schémas, qui englobe celle des données.

Soit M , le formalisme du modèle source M
Soit M ', le formalisme du modèle cible M'
Soit O l'ensemble des opérateurs définis sur les instances d'un schéma de modèle M
Soit O' l'ensemble des opérateurs définis sur les instances d'un schéma de modèle M'
Soit Z_M , l'ensemble des schémas définis dans M
Soit $Z_{M'}$, l'ensemble des schémas définis dans M'
$\forall$ S de Z_M , inst(S) = ensemble de toutes les instances valides de S
la transformation de schémas Σ est définie comme une paire de mapping <T, t>, tels que

$T : Z_M \rightarrow Z_{M'}$ (mapping syntaxique entre les schémas)

$t : inst(S) \rightarrow inst(T(S)) \ \forall$ S de Z_M (mapping entre les instances des schémas)

et il existe $\Sigma^{-1} < T^{-1}, t^{-1} >$ tels que $\forall$ T(S) de $Z_{M'}$,

inst(T(S)) = ensemble de toutes les instances valides de T(S)

$T^{-1} : Z_{M'} \rightarrow Z_M$ (mapping syntaxique inverse entre les schémas)

t^{-1} : $inst(T(S)) \rightarrow inst(T^{-1}(T(S)))$ $\forall$ T(S) de $Z_{M'}$ (mapping inverse entre les instances des schémas)
et tel que $inst(T^{-1}(T(S)) = inst(S)$

T spécifie comment produire le schéma cible à partir du schéma source
t spécifie comment produire les instances du schéma cible à partir de celles du schéma source
T^{-1} et t^{-1} ont des définitions similaires à T et t
t est une combinaison d'opérateurs de O
t^{-1} est une combinaison d'opérateurs de O'

Etant donné que nous serons particulièrement confrontés aux problèmes de transformation de schémas dans des modèles différents, nous mettrons l'accent sur ce type de transformation. Vu la différence d'expressivité des modèles, nous serons parfois confrontés au problème

généralement repris sous l'appellation "**enrichissement sémantique**". En effet, lorsqu'on aura à transformer un schéma décrit dans un modèle M en un schéma décrit dans un modèle M', plus riche en mécanismes de représentation, nous devrons faire face à des constructions plus puissantes pour représenter ces mêmes faits. Cette étape pourra parfois être facilitée par l'existence d'informations extra-schémas, c'est-à-dire des informations fournies par le concepteur du schéma mais non modélisée explicitement.

Ainsi la tâche de traduction peut exiger plus que la traduction syntaxique de schémas et notamment imposer les deux exigences contradictoires suivantes:

- capturer des sémantiques supplémentaires durant la traduction de schémas pour faciliter l'intégration et les mises-à-jour ultérieures

- maintenir seulement les sémantiques existantes car le schéma local n'est pas capable de supporter ces sémantiques supplémentaires.

La prudence s'impose avec le maniement de ce procédé d'enrichissement car les SBD locaux gardent leur autonomie, ce qui signifie que les BD locales ne sont pas modifiées par le processeur de traduction. En imposant au processus de transformation d'être **réversible**, on l'oblige à préserver le contenu du schéma initial.

B. Méthode de transformation de schémas

Chaque modèle dispose de ses propres concepts et règles. Il est toutefois difficile d'établir des correspondances biunivoques entre les concepts de modèles distincts étant donné les sémantiques différentes qui y sont associées.
C'est pourquoi nous travaillerons sur base de structure; cela signifie que nous transformerons une structure (ou un sous-schéma source) en une structure cible.
Nous allons donc utiliser des règles de transformation, celles-ci seront explicitement définies pour deux modèles (source et cible) particuliers et plus précisément entre deux structures de ces mêmes modèles. Ces règles de transformation seront donc constituées des composants suivants : une précondition P représentant la structure source, une postcondition Q représentant la structure cible, les règles de mapping syntaxique (T) entre P et Q et leurs inverses (T^{-1}), les règles de mapping (t) entre les instances de P et Q et leurs inverses(t^{-1}). T permet ainsi de transformer P en Q . Ainsi, une règle transformation est le tuple < P, Q, T, T^{-1}, t, t^{-1}>.

La méthode de transformation(figure 4.12) est donc constituée de deux phases répétitives :

Phase d'identification : l'identification des structures correspondant à une précondition P

Phase d'exécution : l'exécution de la règle correspondante pour obtenir Q, t et t^{-1}

Pour des raisons de rapidité on travaillera généralement au départ des règles de transformations: on prendra la première règle, on identifiera les structures correspondantes dans le schéma source et on exécutera la règle autant de fois qu'il y a eu de structures identifiées.

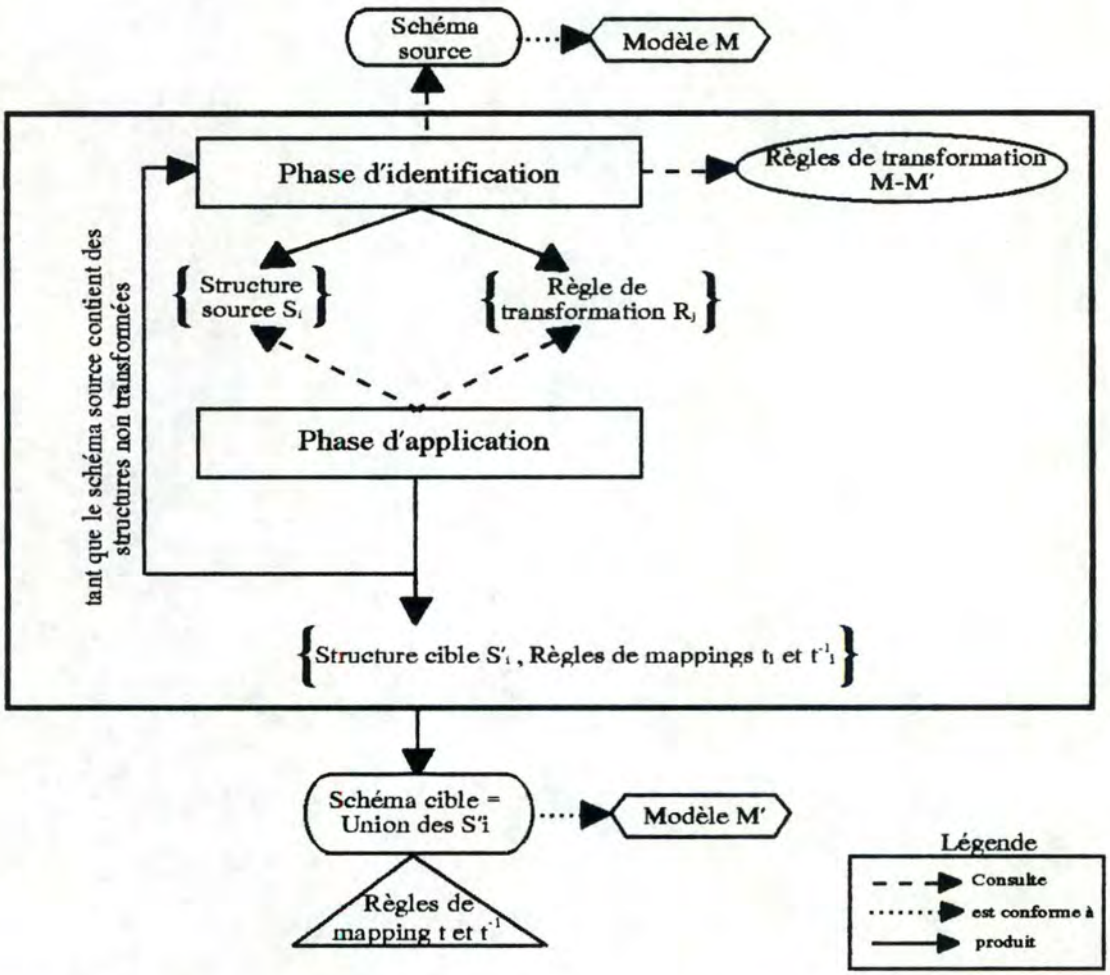


Figure 4.12 : Méthode de transformation de schémas

Nous allons spécifier quelques règles de transformation entre le modèle relationnel et le modèle EA/E et ensuite nous ferons de même pour le modèle Codasyl.

C. Quelques règles de transformation

Nous exposerons les règles de transformation en deux étapes. Tout d'abord nous en donnerons une défintion intuitive à partir d'un **exemple** concret sous forme graphique et ensuite nous fournirons la **définition formelle**.

Nous noterons , pour le modèle EA/E, les concepts de la façon suivante:

TE- E(a_1:d_1(c_{11},c_{12}),...,a_n:d_n(c_{n1},c_{n2}))

pour un TE E, ses attributs $a_1,..,a_n$ leurs domaines $d_1...d_n$ et leurs cardinalités (c_{i1},c_{i2})

TA-A(TE_1 : $r_1(cr_{11},cr_{12})$...,TE_m : $r_m(cr_{m1},cr_{m2})$, a_1:$d_1(c_{11},c_{12})$,...,a_n:$d_n(c_{n1},c_{n2})$)
pour un TA A entre les TE $TE_1..TE_m$, les connectivités (cr_{i1},cr_{i2}) des rôles $r_1..r_m$, les attributs $a_1,..,a_n$ et leurs domaines $d_1...d_n$ ainsi que leurs cardinalités (c_{i1},c_{i2}).

Id(TE) = X , X identifiant du type d'entités TE
Id(TA) = Y , Y identifiant du type d'association TA

Nous utiliserons les notations de l'algèbre EA/E proposées en annexe pour définir les règles de mapping t^{-1}. Nous représenterons une règle de transformation suivant le tableau 4.1.

Règle de transformation R
Précondition P
Postcondition Q
Règle de mapping syntaxique T : élément de la structure Q ↵ élément de la structure P
Règle de mapping syntaxique T^{-1}: élément de la structure Q ↵ élément de la structure P
Règle de mapping entre instances t : valeur d'une occurence d'élément de la structure Q = valeur d'une occurence d' élément de la structure P
Règles de mapping entre instances t^{-1} : valeur d'une occurence d'élément de la structure P = valeur d' une occurence d' élément de la structure Q

Tableau 4.1 : Représentation abstraite d'une règle de transformation

L'expression "**A ↵ B**" exprime le fait suivant :"la structure de données A se déduit de la structure de données B ou autrement dit les structures hétérogènes A et B sont en correspondances"
L'expression "**valeur d'une occurence de l'élément A = valeur d'une occurence de l'élément B**" exprime le fait suivant : "la valeur d'une occurence de l'élément A est égale à la valeur de l'occurence de l'élément B". On parle en terme de ***valeur d'occurence*** étant donné que l'on travaille sur des transformations entre modèles distincts.

C.1. Règles de transformation entre relationnel et EA/E

Trois classes de transformation peuvent être identifiées, chacune d'elle traitant un cas de structure relationnelle. Sous chaque classe, peuvent être regroupées plusieurs règles. Nous ne spécifierons que quelques unes d'entre elles.

Nous allons utilisés les formalismes de représentation suivants:

Soit une **relation** R, ses attributs $a_1, ..., a_n$ leurs domaines respectifs $d_1,...,d_n$
Nous la noterons $R(a_1:d_1,a_2:d_2,...,a_n:d_n)$
Key(R)=a_1 , a_1 attribut **identifiant** de la relation R
$R'.a'_1 \supset R.a_1$ une **contrainte de référence** d'inclusion entre les relations R et R'
$R''.a''_1 = R.a_2$ une **contrainte de référence** d'égalité entre les relations R et R'
Les attributs de références sont en ***italique*** comme a_1
Ses **tuples** seront noté $r_1....r_m$ (**Tuple** = mot spécifique au modèle relationnel pour désigner une occurrence)
Algèbre relationnelle : projection(Π), jointure(*), sélection(σ)

1. Classe de transformation 1

Soit une relation ne reprenant aucun attribut de référence. Cette relation sera transformée en un TE. Tout attribut dont le domaine admet la valeur "null" sera transformé en attribut facultatif. Tout attribut identifiant de la relation sera identifiant du TE correspondant.

Exemple:

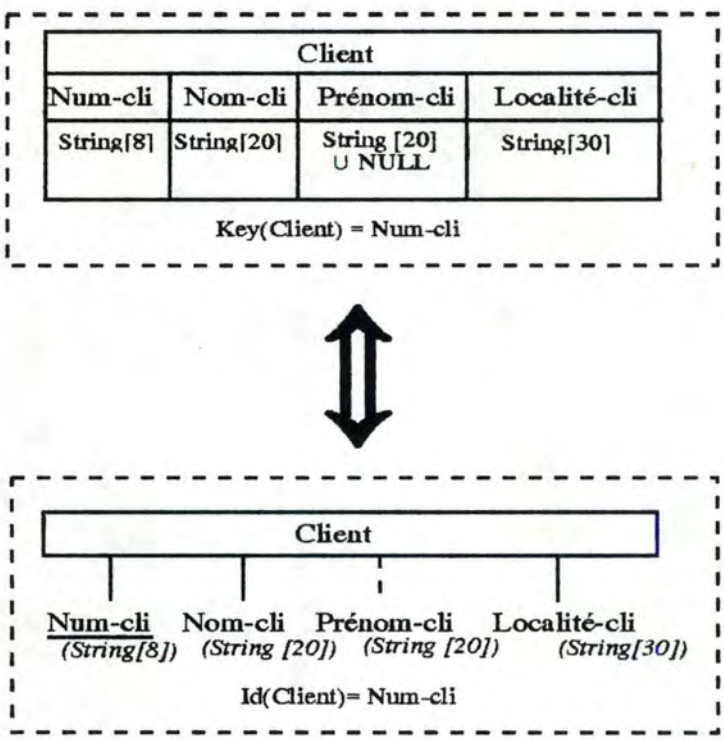
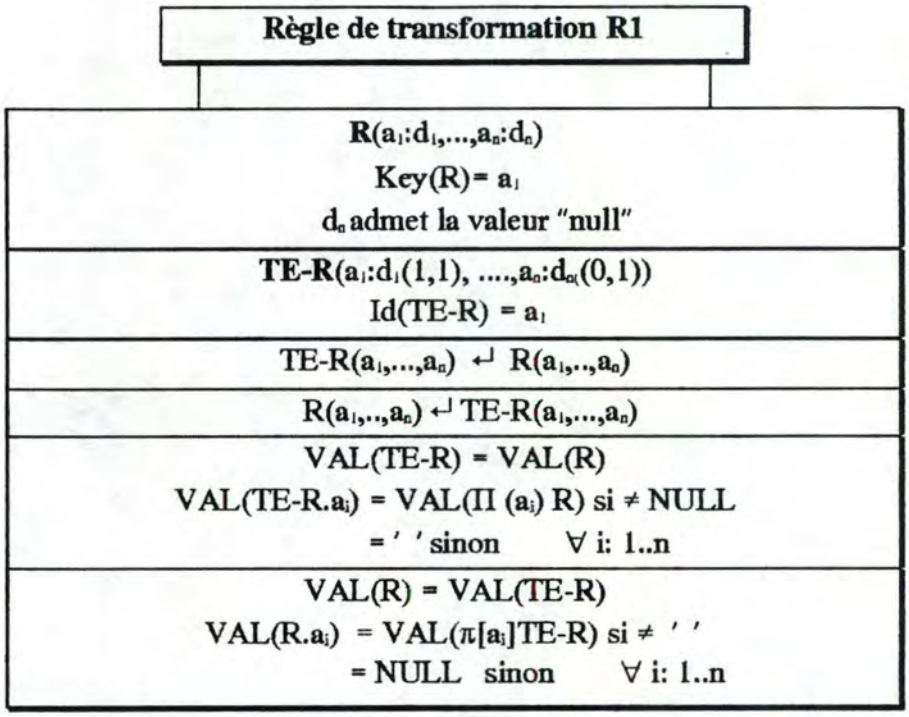


Figure 4.13: Exemple de transformation R1

Règle de transformation R1:



L'expression VAL(A) = VAL(B) signifie que les valeurs de A et de B sont identiques. Cette notation est indispensable étant donné la différence entre les formats des données. Nous éviterons de reprendre cette notation afin d'alléger les écritures.

2. Classe de transformation 2

Soient deux relations dont l'une référence l'autre via **un** des ses attributs de référence et une contrainte référentielle associée.
Cette classe reprendra plusieurs règles suivant le type de contrainte(inclusion, égalité), suivant le nombre d'attribut conjoint de l'attribut de référence, suivant la présence ou non d'un attribut identifiant dans la relation contenant l'attribut de référence.

2.1. Le nombre d'attributs conjoints de l'attribut de référence est quelconque

A chaque relation sera associée un TE de même nom et on définira un TA entre ces deux TE. L'exemple de la figure 2.14 illustre cette situation.

Exemple :

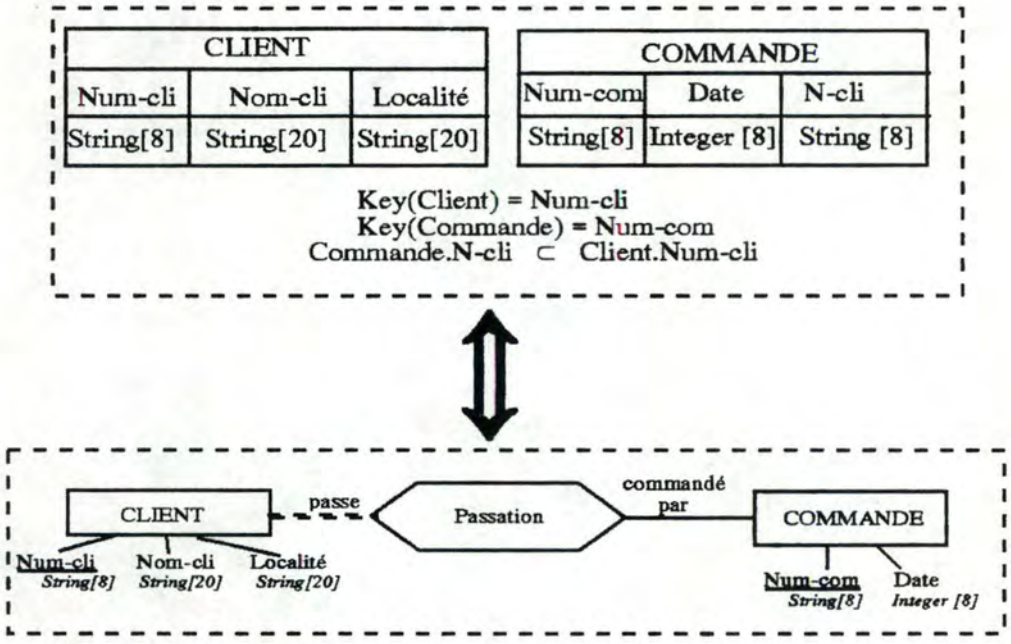


Figure 2.14 : Exemple de transformation R2.1.1

Règle de transformation R2.1.1

Règle de transformation R2.1.1

R1(a_1:d_1,...,a_n:d_n) **R2**(b_1:d'_1,b_2:d_1,..,b_p:d'_p) Key(R1)= a_1 Key(R2)= b_1 $R1.a_1 \supset R2.b_2$
TE-R1 (a_1:d_1(1,1), ...,a_n:d_n(1,1)) Id(TE-R1)= a_1 **TE-R2** (b_1:d'_1(1,1),b_3:d'_3(1,1) ....,bp:d'p(1,1)) Id(TE-R2)= b_1 **TA-R12** (TE-R1 : r_1(0,N), TE-R2 : b_2(1,1))
TE-R1 (a_1,...,a_n) ↵ R1(a_1,...,a_n) **TE-R2** (b_1,b_3,..,b_p) ↵ R2(b_1,b_3,..,bp) **TE-R1** r_1 **TE-R2** ↵ $R1.a_1 \supset R2.b_2$ **TE-R2** b_2 **TE-R1** ↵ $R1.a_1 \supset R2.b_2$ **TA-R12** ↵ R2(b_1,b_2)
R1 (a_1,...,a_n) ↵ TE-R1(a_1,...,a_n) **R2**(b_1,b_3,..,b_p) ↵ TE-R2(b_1,b_3,..,b_p) **R2**(b_2) ↵ b_2

TE-R1(a_1,...,a_n) = R1(a_1,...,a_n) **TE-R2**(b_1,b_3,..,b_p) = Π [b_1,b_3,...b_p] R2 **TE-R2: (TE-R1.a_1 = X)** r_1 **TE-R2** = σ [b_2 = X] R2 ∀ X ∈d_1 **TE-R1: (TE-R2.b_1= Y) b_2 TE-R1** = σ[b_1=Y] R2 ∀ Y ∈d'_1 **TA-R12** (TE-R1 : r_1, TE-R2 : b_2) = Π [b_1,b_2] R2
R1(a_1,..a_n) = TE-R1(a_1,..,a_n) **R2**(b_1,...,b_n) = π [b_1, TA-R12. r_1.b_2, b_3,...,b_p)]TE-R2/r_2 * (TA-R12, TE-R1/r_1)

Les noms attribués aux nouveaux composants(TA-R12,...) sont naturellement susceptibles d'être modifiés afin de les rendre plus significatifs. Par exemple, le rôle "commandé-par" est le nouveau nom du rôle "N-cli".

Règle de transformation R2.1.2.

Si la contrainte de référence était de type égalité au lieu de type inclusion, la connectivité minimale du rôle joué par le TE-R1 sera 1 au lieu de 0. En effet, sur base de l'exemple, cette contrainte signifie que tout Client a au moins une Commande

Règle de transformation R2.1.3.

Si la relation R2 ne contient pas d'autre identifiant que la relation complète alors la cardinalité maximale du rôle joué par le TE-R2 sera N au lieu de 1.

Nous pouvons continuer la panoplie de règles en jouant sur les contraintes en même temps que sur les identifiants...Seules les cardinalités seront changées.

2.2. Cas particulier : Un seul attribut est conjoint de l'attribut de référence et la relation ne dispose pas d'identifiant autre que la combinaison des deux attributs

Le résultat de cette transformation sera un TE auquel est associé un attribut répétitif du même nom que le conjoint de l'attribut de référence.

Exemple:

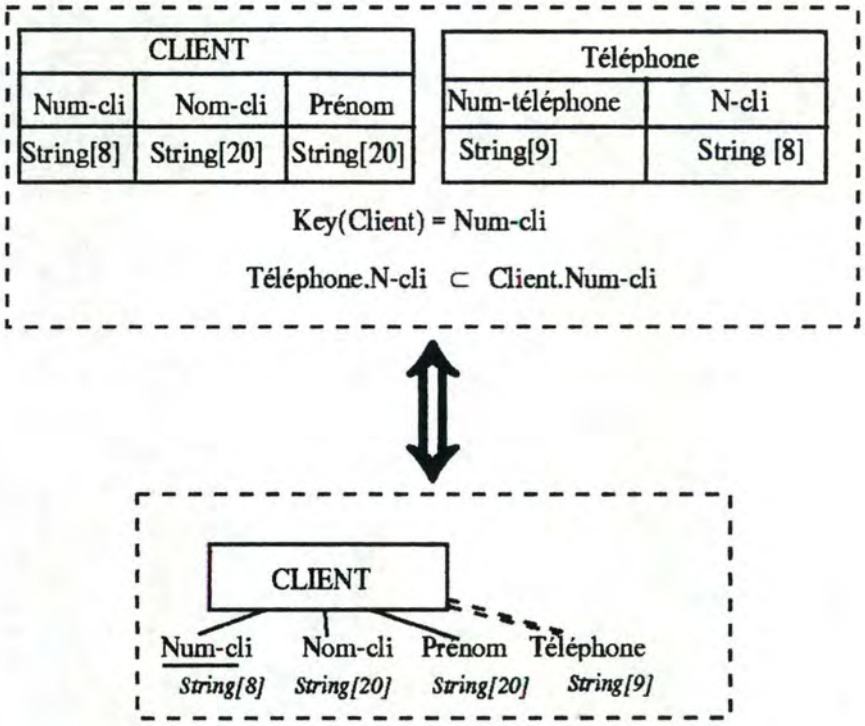


Figure 4.15: Exemple de tranformation R2.2.

Règle de transformation R2.2.

Règle de transformation R2.2

R1(a_1:d_1,...,a_n:d_n) **R2**(b_1:d'_1,b_2:d_1) Key(R1): a_1 R1.a_1 ⊃ R2. b_2
TE-R1(a_1:d_1(1,1), ...,a_n:d_n(1,1), b_1:d'_1(0,n)) Id(TE-R1)=a_1
TE-R1(a_1,..,a_n) ↵ R1(a_1,...,a_n) **TE-R1**(b_1) ↵ R2(b_1)
R1(a_1,...,a_n) ↵ TE-R1(a_1,...,a_n) **R2**(b_1,b_2) ↵ TE-R1(b_1,a_1)
TE-R1(a_1,..,a_n) = R1(a_1,...,a_n) **TE-R1**(b_1) = Π [b_1] σ[b_2=X]R2 ∀ TE-R1.a_1 = X
R1(a_1,...,a_n) = TE-R1(a_1,...,a_n) **R2**(b_1,b_2) = π [b_1^i,a_1] ***TE-I*** avec i:1 ..card(b_1) avec ***TE-I*** = σ[card(b_1)<>0] TE-R1

Nous pouvons formuler les mêmes remarques que précédemment concernant le type de contrainte. Une contrainte d'égalité amènerait la contrainte de cardinalité de l'attribut multivalué à (1,N).

3. Classe de transformation 3

On suppose trois relations dont l'une d'entre elles référence les deux autres. Cette structure sera transformée en deux TE et un TA pour la relation contenant les attributs de référence.

Exemple:

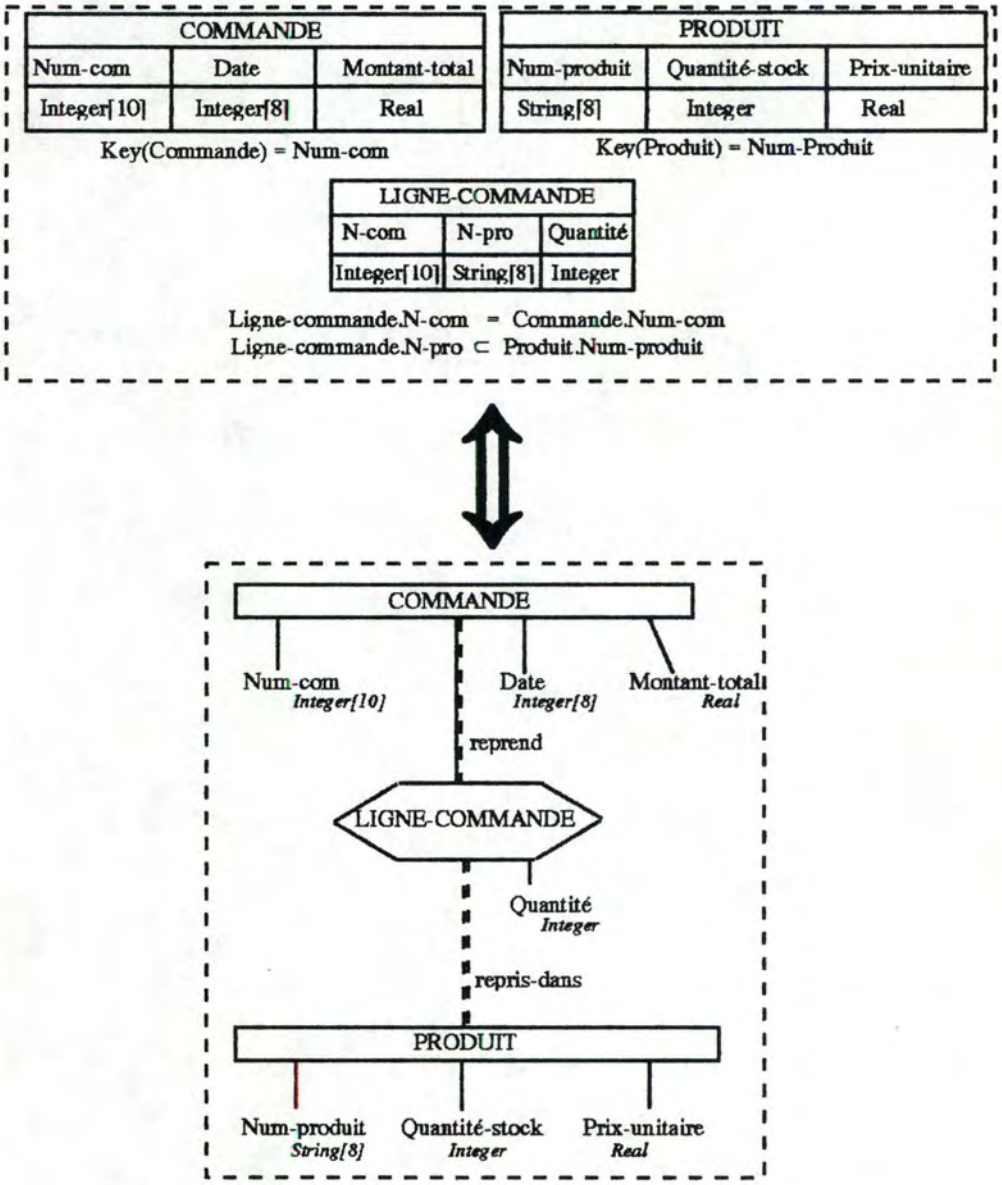


Figure 4.16: Exemple de transformation R3

Règle de transformation R3 :

Règle de transformation R3

R1(a_1:d_1,...,a_n:d_n) Key(R1)= a_1 **R3**(c_1:d''_1,...,c_m:d''_m) Key(R3) = c_1	**R2**(***b_1:d_1,b_2:d''_1***,b_3:d'_3,..,b_p:d'_p) R1.a_1 ⊃ R2. b_1 R3.c_1 ⊃ R2. b_2
TE-R1(a_1:d_1(1,1), ...,a_n:d_n(1,1)) **TE-R3**(c_1:d''_1(1,1),c_2:d''_2(1,1) ....,c_m:d''_m(1,1)) **TA-R2** (TE-R1:b_1(0,N), TE-R3:b_2(0,N), ,b_3:d'_3(1,1),..,b_p:d'_p(1,1)) Id(TE-R1)=a_1, Id(TA-R3)= c_1	
TE-R1(a_1, ...,a_n) ↵ R1(a_1,...,a_n) **TE-R3**(c_1 ....,c_m) ↵ R3(c_1,...,c_m) **TA-R2** (TE-R1:b_1,TE-R3:b_2, b_3,..,b_p) ↵ R2(b_1,b_2,b_3,..,b_p) **TE-R1** b_1 **TE-R3** ↵ R1.a_1 ⊃ R2. b_1 **et** R3.c_1 ⊃ R2. b_2 **TE-R3** b_2 **TE-R1** ↵ R1.a_1 ⊃ R2. b_1 **et** R3.c_1 ⊃ R2. b_2	
R1(a_1,...,a_n) ↵ TE-R1(a_1, ...,a_n) **R3**(c_1,...,c_m) ↵ TE-R3(c_1 ....,c_m) **R2**(b_1,b_2,...,b_p) ↵ TA-R2 (TE-R1:b_1,TE-R3:b_2, b_3,..,b_p)	
TE-R1(a_1, ...,a_n) = R1(a_1,...,a_n) **TE-R3**(c_1 ....,c_m) = R3(c_1,...,c_m) **TA-R2** (TE-R1:b_1,TE-R3:b_2, b_3,..,b_p) = R2(b_1,b_2,b_3,..,b_p) **TE-R3:(TE-R1.a_1=X)** r_1 **TE-R3** = Π [b_2] σ[b_1=X] R2 ∀ X ∈d_1 **TE-R1:(TE-R3.c_1=Y)** r_3 **TE-R1** = Π [b_1] σ[c_1=Y] R2 ∀ Y ∈d''_1	
R1(a_1,...,a_n) = TE-R1(a_1, ...,a_n) **R3**(c_1,...,c_m) = TE-R3(c_1 ....,c_m) **R2**(b_1,b_2,b_3,..,b_p) = π [a_1,TA-R2.b_2.c_1,TA-R2. TA-R2.b_1.(b_3,...,b_p)] TE-R1/b_1 * (TA-R2, TE-R2/b_2)	

Changer les contraintes de référence conduira à modifier les connectivité des rôles comme il en a déjà été question.

A présent analysons les règles de transformation pour le modèle Codasyl.

C.2. Règles de transformation entre Codasyl et EA/E

Le modèle réseau Codasyl offre les concepts de record, d'attribut(décomposable ou atomique, simple ou répétitif) et la notion de SET qui permet de regouper sous un agrégat un record d'un type et 0,1 ou plusieurs record d'un autre type.

1. Transformation d'un type de record en un type d'entités

A un type de record, on fera correspondre un type d'entités de même nom dans le modèle EA/E. Nous reprendrons les mêmes noms et domaines pour les attributs. Notons que la cardinalité minimale de tout attribut sera 1. Pour les attributs répétitifs, la cardinalité maximale sera N en l'absence d'un nombre précis.

Exemple:

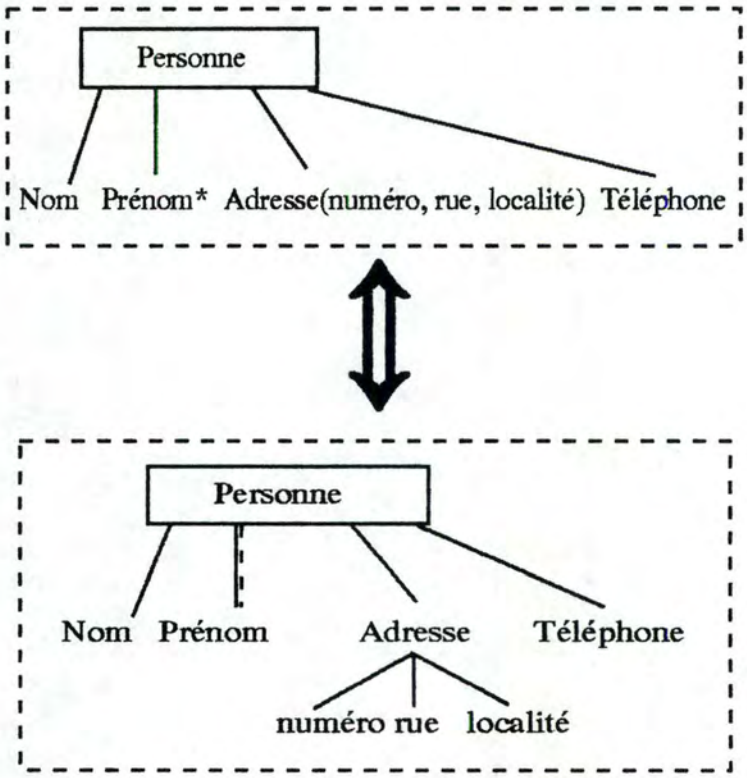
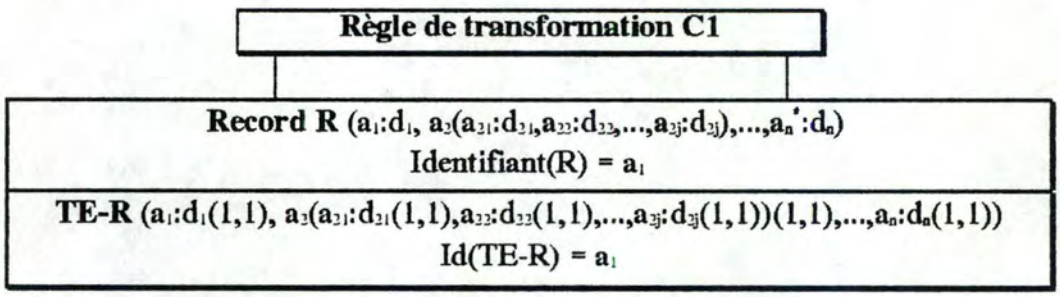


Figure 4.17. : Exemple de transformation C1

Règle de transformation C1



Les règles de mapping apparaissent évidentes étant donné la simplicité de la transformation.

2. Transformation d'un type de SET en un type d'association

Un type de SET défini entre deux types de record sera transformé en un type d'association entre les TE correspondants aux types de record. Etant donné les propriétés d'un type de SET, les connectivités des rôles seront (0-N)(0,1). Toutefois, il existe des **clauses particilères** en Codasyl qui permettent d'obliger l'appartenance d'un type de record à un set. Ainsi les connectivités des rôles pourraient être (1,N) et/ou (1,1).Nous ne donnerons la règle que pour le cas général. Il est toutefois aisé de la modifier afin de prendre en compte cette clause particulière(cfr Illustration Chapitrte5).
On dénomme le type de record CLIENT(Figure 4.18) comme OWNER et le type de record COMMANDE(Figure 4.18) comme MEMBER du SET CC.

Exemple :

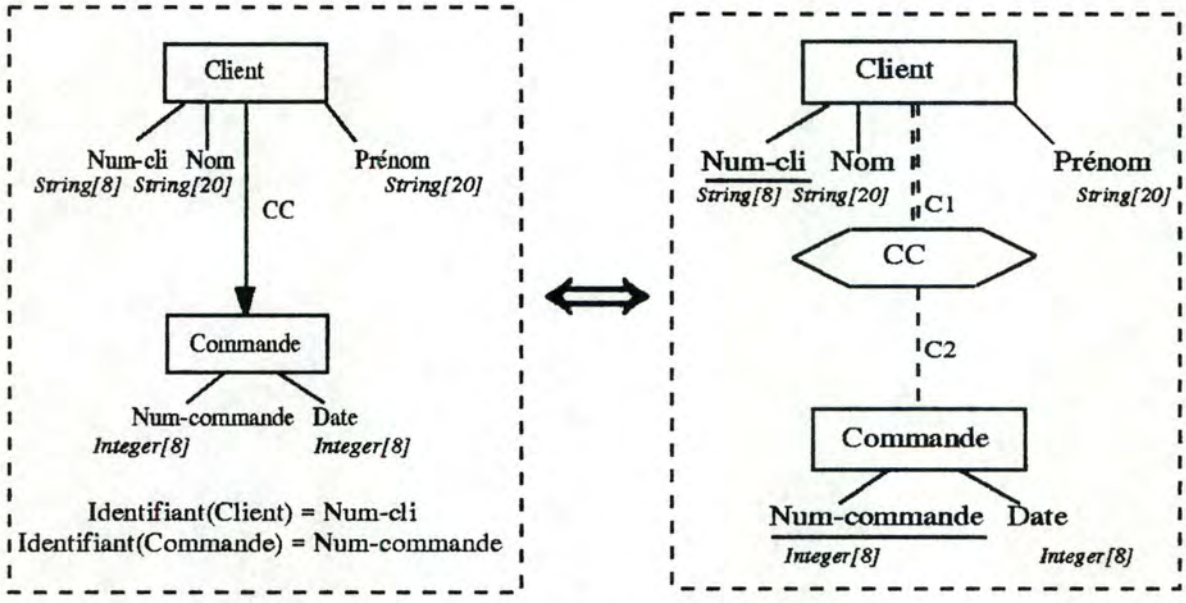
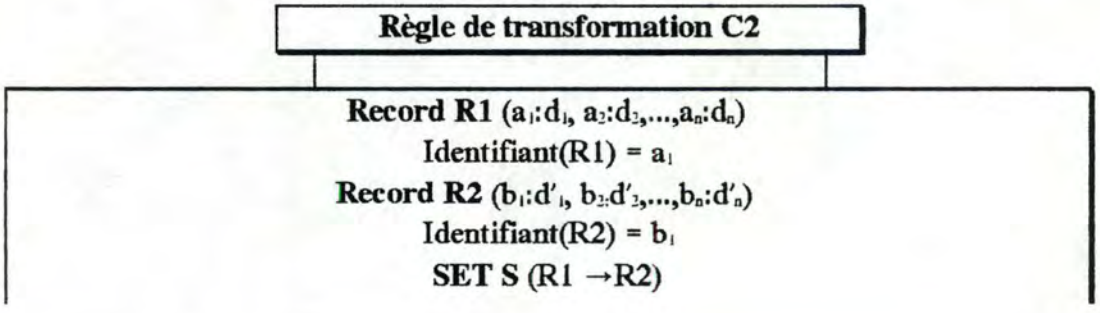


Figure 4.18: Exemple de transformation C2

Règle de transformation C2:



TE-R1 (a_1:d_1(1,1), a_2:d_2(1,1),...,a_n:d_n(1,1)) Id(TE-R1) = a_1 **TE-R2** (b_1:d'_1(1,1), b_2:d'_2(1,1),...,b_n:d'_n(1,1)) Id(TE-R2) = b_1 **TA-S** (TE-R1:s_1(0,N),TE-R2:s_2(0,1))
TE-R1(a_1, a_2,...,a_n) ↵ Record R1 (a_1, a_2,...,a_n) **TE-R2**(b_1, b_2,...,b_n) ↵ Record R2 (b_1, b_2,...,b_n) **TA-S** (TE-R1:s_1,TE-R2:s_2) ↵ SET S **TE-R1** s_1 **TE-R2** ↵ Record R1 S Record R2 **TE-R2** s_2 **TE-R1** ↵ Record R2 S Record R1
Record R1(a_1, a_2,...,a_n) ↵ TE-R1(a_1, a_2,...,a_n) **Record R2** (b_1, b_2,...,b_n) ↵ TE-R2(b_1, b_2,...,b_n) **SET S** ↵ TA-S (TE-R1:s_1,TE-R2:s_2)
TE-R1(a_1, a_2,...,a_n) = Record R1(a_1, a_2,...,a_n) **TE-R2**(b_1, b_2,...,b_n) = Record R2 (b_1, b_2,...,b_n) **TA-S** (TE-R1:s_1,TE-R2:s_2) = (OWNER,MEMBER)OF SET S **TE-R2 : TE-R1** s_1 **TE-R2** = MEMBER of SET S **TE-R1 : TE-R2** s_2 **TE-R1** = OWNER of SET S
Record R1(a_1, a_2,...,a_n) = TE-R1(a_1, a_2,...,a_n) **Record R2** (b_1, b_2,...,b_n) = TE-R2(b_1, b_2,...,b_n) **MEMBER of SET S** = π [TA-S. s_2] TE-R1/s_1 *(TA-S, TE-R2/s_2) **OWNER of SET S** = π [TA-S . s_1] TE-R2/s_2 *(TA-S, TE-R1/s_1)

3. Transformation de deux types de SET en un type d'association

Lorsqu'on a deux types de SET qui ont le même type de record comme MEMBER et que ce dernier ne contient pas d'attribut alors on peut transformer cette structure en un type d'association dont les rôles sont (0,N)(0,N).

Exemple :

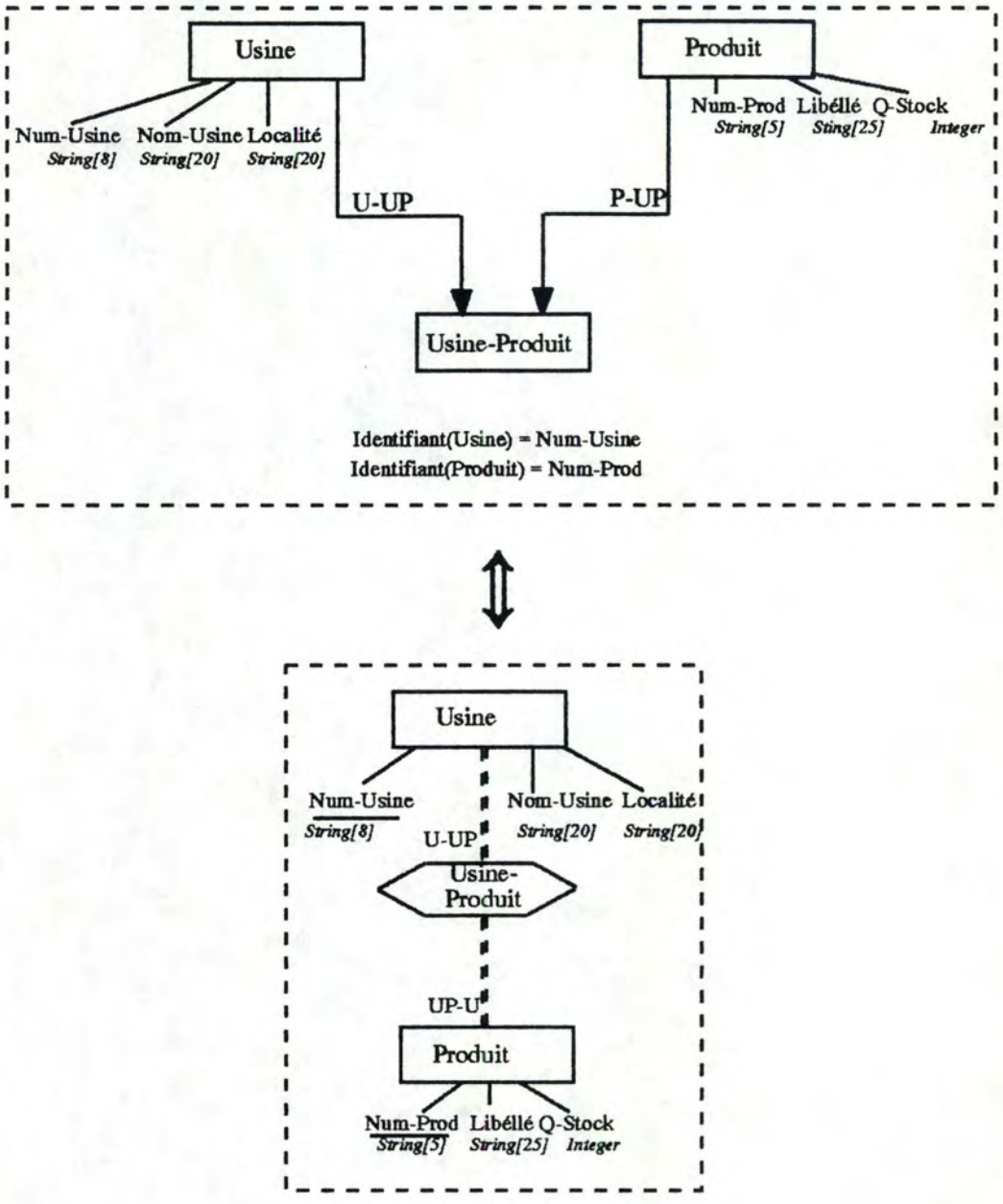


Figure 4.19. : Exemple de transformation C3

Règle de transformation C3 :

Règle de transformation C3

Record R1 (a_1:d_1, a_2:d_2,...,a_n:d_n)
Identifiant(R1) = a_1
Record R2 (b_1:d'_1, b_2:d'_2,...,b_n:d'_n)
Identifiant(R2) = b_1
Record R3 ( )
SET S1 (R1 → R3)
SET S2 (R2 → R3)

TE-R1(a_1:d_1(1,1), a_2:d_2(1,1),...,a_n:d_n(1,1)) Id(TE-R1) = a_1 **TE-R2**(b_1:d'_1(1,1), b_2:d'_2(1,1),...,b_n:d'_n(1,1)) Id(TE-R2) = b_1 **TA-R3**(TE-R1:S_1(0,N), TE-R2:S_2(0,N))
TE-R1(a_1,a_2,...,a_n) ↵ Record R1 (a_1,a_2,...,a_n) **TE-R2** (b_1, b_2,...,b_n) ↵ Record R2 (b_1, b_2,...,b_n) **TA-R3**(TE-R1:S_1, TE-R2:S_2) ↵ Record R3( ) **TE-R1 S1 TA-R3** ↵ Record R1 S1 Record R3 **TE-R2 S2 TA-R3** ↵ Record R2 S2 Record R3
Record R1 (a_1,a_2,...,a_n) ↵ TE-R1 **Record R2** (b_1, b_2,...,b_n) ↵ TE-R2 **Record R3** ( ) ↵ TA-R3 **SET S1** ↵ TE-R1 S1 TE-R3 **SET S2** ↵ TE-R2 S2 TE-R3
TE-R1(a_1,a_2,...,a_n) = Record R1 (a_1,a_2,...,a_n) **TE-R2** (b_1, b_2,...,b_n) = Record R2 (b_1, b_2,...,b_n) **TA-R3**(TE-R1:S1, TE-R2:S2) = Record R3( ) **TE-R2 : TE-R1 S1 TE-R2** = OWNER of S2 via (MEMBERS of SET S1) **TE-R1: TE-R2 S2 TA-R1** = OWNER of S1 via (MEMBERS of SET S2)
Record R1 (a_1,a_2,...,a_n) = TE-R1 (a_1,a_2,...,a_n) **Record R2** (b_1, b_2,...,b_n) = TE-R2 (b_1, b_2,...,b_n) **Record R3** ( ) = TA-R3(TE-R1:S1, TE-R2:S2) **MEMBER of SET S1** = π [TA-R3 .TA-R3] TE-R1/S1 *(TA-R3, TE-R2/S2) **OWNER of SET S1** = TE-R1: TA-R3 S1 TE-R1 **MEMBER of SET S2** = π [TA-R3.TA-R3] TE-R2/S2 *(TA-R3, TE-R1/S1) **OWNER of SET S2** = TE-R2: TA-R3 S1 TE-R2

Ce que nous venons de faire pour les modèles relationnel et Codasyl, il faudra le faire pour tous les modèles des SBD locaux afin de pouvoir traduire tous les schémas dans le modèle EA/E.
A présent nous allons analyser le problème de tranformation des requêtes.

4.3.3. Transformation de requêtes

A. Définition intuitive et formelle

A chaque modèle est associé un langage de manipulation. Vu la diversité des modèles, nous sommes confrontés à une multitude de langages de manipulation. Se posera donc le problème de la traduction dans un **langage** *cible* de requêtes exprimées dans un langage *source*. A cela vient s'ajouter le problème de la traduction des **arguments** de ces requêtes. En effet, une requête est toujours en rapport avec un schéma, structuré dans un certain modèle.

Les préconditions à l'exécution de cette transformation sont doubles: la transformation des schémas doit être achevée et les règles de mapping entre les schémas source et cible doivent être disponibles pour l'exécution du processsus de transformation de requêtes.

Ainsi, la transformation d'une requête R définie dans un langage L en fonction des concepts d'un schéma S décrit dans un modèle M consiste à fournir une requête R' dans un langage L' en fonction des concepts d'un schéma S' exprimé dans un modèle M'. Intuitivement, cette transformation doit être préservatrice de sémantique en ce sens que le résultat fourni par la requête R' doit être le même que celui qui aurait été fourni par la requête R si les données avaient été stockées au même niveau que le schéma S.
Formellement, nous définirons la transformation de requêtes comme suit :

Soient

M, formalisme du modèle source,
M', le formalisme du modèle cible,
L, le langage source associé à M,
L', le langage cible associé à M',
S, le schéma source décrit en M,
S', le schéma cible décrit en M',
T , les règles de mapping entre S et S',
Z_S^L, l'ensemble des requêtes en L sur S,
$Z_{S'}^{L'}$, l'ensemble des requêtes en L' sur S',

$\forall R \in Z_S^L$, la transformation de requête Θ est défini comme :

$$\Theta : Z_S^L \rightarrow Z_{S'}^{L'} : R' = \Theta(R) \quad \text{et } \Theta = < \varphi, T >$$

avec φ les correspondances entre les structures des langages L et L' telles que à toute structure du langage L corresponde une structure du langage L' ou

$$\varphi : L \rightarrow L' : s' = \varphi(s) \quad \forall s \text{ structure du langage } L$$

Dans une transformation de requête, on distingue donc deux parties: la primitive(ou opérateur) et les arguments. Les correspondances entre arguments de schémas différents ont été établis lors de la transformation du schéma (le composant T de cette transformation). Par contre les correspondances entre primitives sont à développer. Avant tout, nous allons établir une typologie des transformations de requêtes, ce qui nous permettra de mieux percevoir les problèmes.

B. Typologie des transformations de requêtes

Suite aux deux types de langages les plus utilisés, les langages assertionnels et procéduraux, on distingue quatre type de processus de transformation de requêtes:

TR-1 : Traduction dans un langage *assertionnel* LA_2 d'une requête exprimée dans un langage *assertionnel* LA_1

TR-2 : Traduction dans un langage *procédural* LP_2 d'une requête exprimée dans un langage *procédural* LP_1

TR-3 : Traduction dans un langage *assertionnel* LA d'une requête exprimée dans un langage *procédural* LP

TR-4 : Traduction dans un langage *procédural* LP d'une requête exprimée dans un langage *assertionnel* LA

TR-1: LA_1 -> LA_2 ($LA_1 \neq LA_2$)

L'exemple classique est celui de la traduction d'une requête SQL en une requête QUEL pour les modèles relationnels. Ce type de traduction peut paraître assez simple mais peut néanmoins poser de nombreux problèmes. Notamment lorsqu'il est impossible d'établir des correspondances 1-1 entre les opérations des deux langages. Ce type de processus doit être développé en pleine connaissance de la sémantique des opérations des deux langages. Un autre problème se pose plus particulièrement pour les requêtes imbriquées et les opérations particulières d'agrégation.
Nous serons confrontés à ce type de transformation que ce soit d'un langage assertionnel quelconque vers le langage L-EA/E ou inversément.

TR-2: LP_1 -> LP_2 ($LP_1 \neq LP_2$)

Cette transformation est des plus complexes à mettre en oeuvre mais heureusement pour nous est très rare d'utilisation. En effet, vu les recommandations concernant le choix du modèle canonique et de son langage, ce genre de situation ne se présente pas.

TR-3: LP -> LA

Cette transformation est complexe à réaliser. Elle sera néanmoins utile dans la mesure où l'on souhaite permettre aux utilisateurs de définir des schémas externes dans le modèle qui leur est familier. Nous devrons être en mesure de traduire des requêtes Codasyl en L-EA/E,...

TR-4: LA -> LP

L'exemple typique est la traduction d'une requête SQL en un programme Codasyl. Dans notre contexte, nous aurons besoin de traduire des requêtes L-EA/E en requêtes Codasyl.

La figure 4.20 donne un exemple de transformation de requêtes basé sur les figures 4.14 et 4.18.

Figure 4.14: "Donner l'ensemble des Commandes(Numéro et date) du Client dont le Num-cli = "KP25L261"

En L-EA/E	En SQL
SELECT Num-com, Date FROM COMMANDE,CLIENT WHERE Commande commandé-par Client AND Num-cli="KP25L261"	Select Num-com, Date From Commande Where Ncli="KP25L261"

Figure 4.18: "Donner l'ensemble des Commandes(Numéro et date) du Client dont le Num-cli = "KP25L261"

En L-EA/E	En Codasyl DML
SELECT Num-commande, Date FROM CLIENT,COMMANDE WHERE Commande C2 Client AND um-cli="KP25L261"	Error = 0 Move "KP25L261" to Num-cli of Client Find Client record If error <> 0 display "Client inconnu" else Find first Commande record of CC if error <> 0 Display Num-commande,Date Perform Traiter-suivant until error <>0 else Display"Pas de commande" Traiter-suivant Find next commande record of CC if error <> 0 Display Num-commande, Date

Figure 4.20 :Exemple de transformation de requêtes

C. Méthode de transformation de requêtes

Pour les transformations entre langages assertionnels, la méthode consiste à définir une **matrice de transformation** pour les langages concernés. Cette matrice a deux dimensions : l'une reprend l'ensemble des opérations ou primitives offertes par le langage et l'autre l'ensemble des structures sur lesquelles portent les primitives. A l'intersection de chacune des ligne et colomne de la matrice, on trouvera la règle à appliquer. L'avantage de cette méthode est sa facilité d'extension. Il est en effet aisé d'ajouter des règles si l'on ajoute des nouvelles primitives ou nouveaux concepts de modélisation.

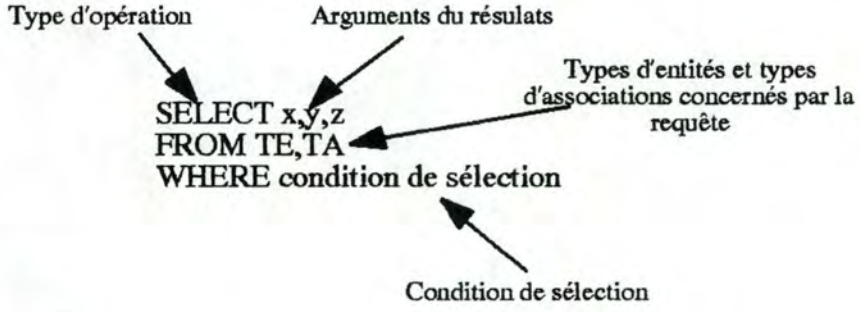
Pour les transformations entre langage assertionnel et procédural, nous ferons appel aux algorithmes de génération automatique de code. Par exemple, une jointure entre deux relations

(en relationnel) sera transformée en deux boucles imbriquées, chacune d'entre elles parcourant l'équivalent Codasyl d'une des relations. En règle générale, cela signifie que pour une opération assertionnelle impliquant une relation, on définira un algorithme qui générera automatiquement le code correspondant.

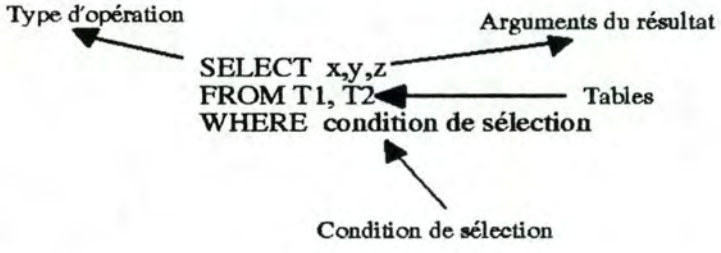
On se limitera dans le cadre de ce mémoire à fournir quelques exemples de transformation de requêtes de type TR-1

D. Quelques transformations de requêtes de type TR-1

Si on regarde une requête type en L-EA/E, on remarque qu'elle peut être découpée en trois parties:



En SQL, les requêtes doivent en plus préciser les tables relationnelles concernées comme suit :



Malgré les ressemblances entre les structures de ces requêtes de sélection, la transformation n'est pas triviale étant donné que le modèle EA/E manipule des rôles et pas le modèle relationnel.
Ainsi, pour transformer une requête de type L-EA/E en une requête SQL, on opérera comme suit: les arguments du résultat et les tables de la requête cible se déduisent aisément par les règles de mapping T. En ce qui concerne la condition de sélection, il faut être plus attentif étant donné que les modèles EA/E et relationnel sont très différents.
Prenons l'exemple de la figure 4.14 et définissons la requête L-EAE suivante:

SELECT Num-com, Date
FROM Commande,Client
WHERE Num-cli="XYPI2513" AND Client passe Commande

Sur base des règles de mapping T (que l'on déduit aisément de la règle formelle R2.1.1), on obtient la requête suivante:

SELECT Num-com, Date
FROM COMMANDE, CLIENT
WHERE Num-cli = "XYPI2513" AND N-cli = Num-cli

Nous constatons de suite que cette requête peut être simplifiée comme suit:

SELECT Num-com, Date
FROM COMMANDE
WHERE N-cli = "XYPI2513"

Pour la même figure, donnons l'exemple suivant:

SELECT Nom-cli, Localité
FROM Client, Commande
WHERE Num-com = "CCCC2153" AND Commande commandée-par Client

qui est transformée en:

SELECT Nom-cli, Localité
FROM Client, Commande
WHERE Num-com = "CCCC2153" AND Num-cli=N-cli

et cette dernière ne peut plus être simplifiée.

Par ces exemples, nousa vons montré l'importance des règles de mapping T. Nous avons également mis l'accent sur la nécessité d'éliminer les éléments inutiles dans les requêtes cibles avant de les exécuter. Ces facteurs vont jouer un rôle important sur le temps de réponse de ces requêtes dont il sera question dans la section 4.5. La qualité de la transformation est primordiale pour obtenir des temps de réponse acceptables.
Avant de clôturer ce paragraphe nous allons fournir quelque indication pour le transformation des opérations de mise-à-jour. Ces dernières s'avèrent généralement difficile étant donné la présence des TA et rôles associés aux TE.

Sur base de la figure 4.14, si l'on souhaite supprimer une association "passation" entre un "client "et une de ses "commandes", cela implique directement la suppression d'une entité "commande" afin de respecter les contraintes. Toutefois cette suppression d'entité n'est pas programmée par l'utilisateur. Et pourtant cette suppression d'association couplée avec celle d'une entité "commande" sera transfomée en une seule requête de suppression d'un tuple de la relation "Commande". On constate donc que le problème de transformation de requêtes est un problème complexe qui nécessiterait plus de réflexion. Etant donné l'étendue du problème des BDH, il est impossible, dans le cadre d'un mémoire, de s'attaquer à tous les problèmes. C'est pourquoi, on s'est contenté de présenter le problème et de fournir quelques remarques.

4.3.4. Bases de données hétérogènes et "reverse engineering"

Jusqu'à présent on avons émis fait l'hypothèse implicite que tout SBD local disposait d'un schéma conceptuel (relationnel, fonctionnel,...)de la base de données qu'il stocke. Or cette hypothèse n'est pas toujours vérifiée dans la réalité. Nombreuses sont les bases de données où les seuls renseignements dont on dispose sont les textes d'implantation de la base de données(définition des fichiers Cobol,...). La tâche du reverse engineering est justement de reconstruire automatiquement le schéma conceptuel sur base de ces spécifications techniques des bases de données. Il s'agit d'un problème très complexe que nous citons au passage afin de montrer que le problème des BDH peut s'avérer encore plus difficile à résoudre que ce qui est présenté dans le cadre de ce mémoire.

4.4. Problèmes d'intégration

4.4.1. Introduction

Afin d'offrir une vue unique et transparente sur l'ensemble des SBD locaux, nous avons décidé de définir un schéma fédéré c'est-à-dire un schéma reprenant l'ensemble des schémas exportés (parties des schémas composants). La première étape vers la définition de ce schéma a été celle de l'homogénéisation des schémas locaux dans un formalisme unique, celui du modèle canonique, conduisant ainsi aux schémas composants. En émettant l'hypothèse que les schémas exportés ont été définis ou qu'ils sont identiques aux schémas composants, nous allons passer à l'étape suivante, celle de la définition du schéma fédéré. Il s'agira de définir un schéma unique sur base des schémas exportés qui, rappelons le, sont exprimés dans un formalisme unique. C'est pourquoi nous parlerons d'**intégration de schémas** pour désigner cette étape. Le schéma résultant de cette étape et que nous appellerons en toute généralité *schéma intégré* devra vérifier certaines caractéristiques que nous préciserons. Etant donné que les SBD locaux ont été définis de façon indépendante et dans des modèles différents, lors de cette étape on devra faire face à des conflits de représentation entre les différents schémas exportés. Nous exposerons donc une typologie des conflits susceptibles d'être rencontrés. Nous exposerons ensuite différentes approches de solutions et leurs critiques. Vu que l'intégration de schémas n'a pour objectif que de fournir un schéma et non une base de données, il conviendra de définir des **règles de mappings** entre le schéma fédéré et les schémas exportés. Ces mappings serviront ultérieurement pour le traitement des requêtes. Nous aborderons donc également le problème de la définition de ces règles de mapping. Finalement, nous analyserons le problème issu du traitement des requêtes et qui consiste à **intégrer les données** fournies par les différents SBD locaux. La résolution de ce problème est essentielle pour fournir à l'utilisateur un résultat unique et transparent. Rappelons que l'intégration des données se fera sur base du modèle canonique et que nous devrons passer au préalable par une phase de transformation de données, comme elle a été proposée à la section 4.3.2. Pour mieux comprendre la corrélation existant entre ces trois problèmes, examinons la figure 4.21.

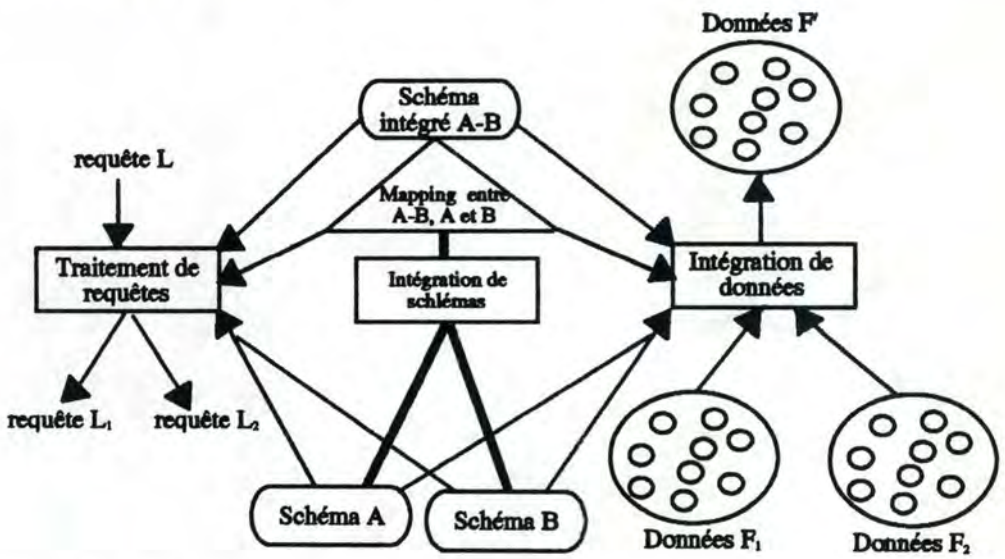
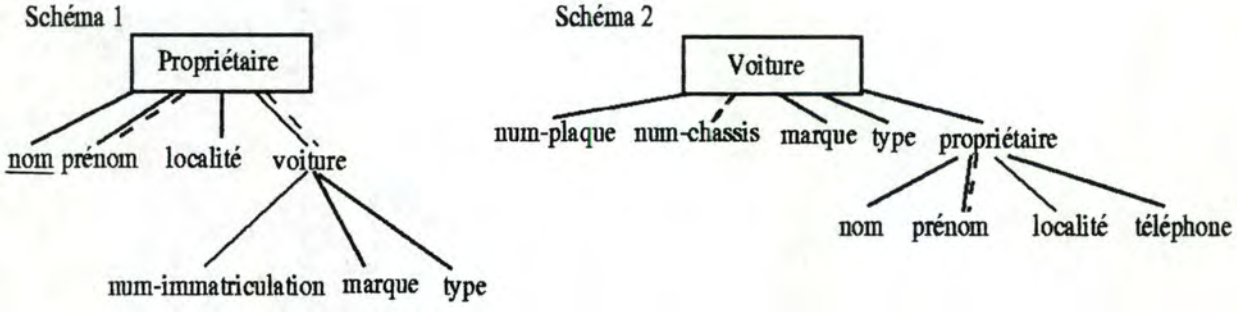


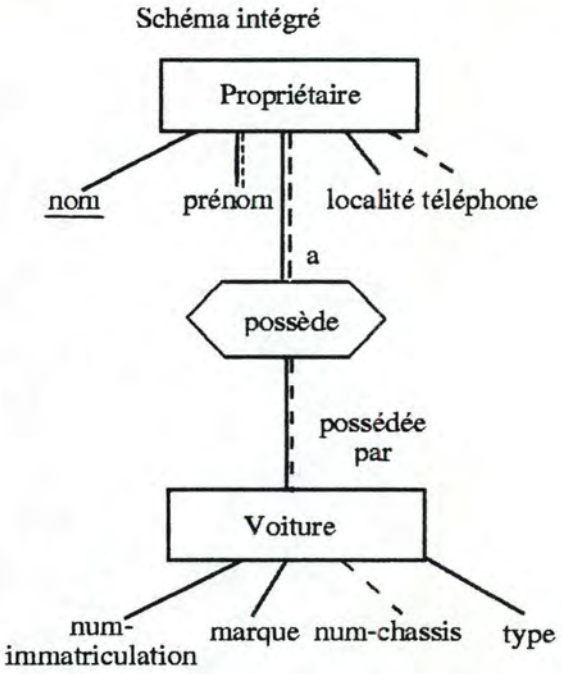
Figure 4.21. : Corrélation entre problèmes d'intégration de schémas, de données et mapping

Nous remarquerons que la phase d'intégration de schémas est une opération "**one-shot**" alors que la phase d'intégration de données est une opération "**temps réel**" qui doit être faite à chaque requête. La phase d'intégration est une tâche complexe et qui peut prendre beaucoup de temps. Mais comme il s'agit d'une opération one-shot, nous estimons que le problème ne se situe pas à ce niveau-là, ce qui n'est pas le cas pour l'intégration de données. C'est pourquoi, les règles de mappings sont d'une importance considérable.
La figure 4.22. donne un exemple d'intégration de schémas et de données auquel nous pourrions être confronté. Pour faciliter la lecture, nous considérerons que les données sont présentes avec les schémas à intégrer, ce qui ne sera pas le cas dans la réalité.



Propriétaire (Schéma 1)			
nom	prénom	localité	voiture(num-imma, marque, type)
Dupont	Jean, Gérard	Namur	(KBS.112, Peugeot, 405), (JAS.100, Renault, Espace)
Durant	Jacques	Bruxelles	(ACA.050, VW, Golf)

Voiture (Schéma 2)				
num-plaque	num-chassis	marque	type	propriétaire(nom,prén.,loc.,téléph.)
KBS.112	XIBFR25125	Peugeot	405	Dupont, (Jean,Gérard),Namur, 125
AVF.205	FGRDTR854	Audi	80	Smith, (Bernard), Tournai, 263



Requête sur sch. Intégré: Donner l'ensemble des voitures			
num-immatr.	num-chassis	marque	type
KBS.112	XIBFR25125	Peugeot	405
AVF.205	FGRDTR854	Audi	80
JAS.100		Renault	Espace
ACA.050		VW	Golf

Figure 4.22 : Exemple d'intégration de schémas et de données

4.4.2. Caractéristiques attendues d'un schéma intégré

Avant de préciser ces caractéristiques, rappelons le contexte dans lequel nous travaillons. Il s'agit de celui des bases de données hétérogènes, ce qui signifie que l'on est face à des bases de données existantes. Ce contexte ne doit pas être confondu avec celui d'intégration de vues de bases de données que l'on rencontre en conception de bases de données. En conption de BD, on est également confronté au problème d'intégration de schémas mais ces derniers ne supporte encore aucune base de données. Ainsi, ces deux problèmes d'intégration quoique quasi identiques d'un point de vue extérieur sont à distinguer, notamment dans la qualité du résultat à fournir, c'est-à-dire le schéma intégré. Cette distinction porte sur le fait que dans le contexte des BDH, il doit toujours être possible de se référer aux schémas des bases de données locales tandis que dans le contexte de conception de BD, on ne demande pas de pouvoir faire référence par la suite aux schémas d'origine. Cette distinction peut conduire à la création de schémas intégrés différents. Dans l'intégration de vue (contexte de conception), on pourra aboutir à un schéma fort différent par rapport aux schémas d'origine, ce qui n'est pas le cas dans le contexte des BDH.

Dans le cadre des BDH, nous pouvons définir trois caractéristiques que doit respecter le schéma intégré: la complétude, la minimalité et la compréhensibilité. Nous donnerons également une exigence concernant les règles de mapping.

A. La complétude

Le schéma intégré doit contenir tous les concepts repris dans chaque schéma exporté. On évitera donc de regrouper sous un seul élément(TE, TA ou attribut) ce qui était présenté à l'aide de plusieurs éléments. Par exemple, si l'on retrouve dans un schéma exporté les concepts d'homme et de femme alors que dans un autre schéma exporté on y retrouve uniquement le concept d'humain, on préférera conserver les trois concepts plutôt que de garder uniquement une des deux visions. Les mécanismes d'abstraction offerts par le modèle EA/E ont été définis de telle sorte à faciliter la résolution de telles problèmes. Pour le cas présent, on utiliserait la relation de généralisation.

B. La minimalité

Un même concept apparaissant dans plusieurs schémas exportés ne doit figurer qu'une seule fois dans le schéma intégré, afin d'éviter la redondance au sein du schéma intégré. Ainsi, si l'on retrouve le concept de "Client" dans plusieurs schémas exportés, celui-ci ne sera repris qu'une seule fois dans le schéma intégré.
Il faudra notamment veiller à éliminer les relations qui peuvent se déduire d'autres relations. Exemple des relations is-a : Un doctorant est un (is-a) assistant, un assistant est une(is-a) personne. Inutile de préciser qu'un doctorant est une(is-a) personne, vu la propriété de transitivité des relations de généralisation.

C. La compréhensibilité

Le schéma doit être facile à comprendre pour l'utilisateur potentiel. Par conséquent, face à plusieurs solutions possibles pour résoudre un problème d'intégration, on choisira celle qui est la plus compréhensible. Le choix d'un mécanisme d'abstraction agit sur la compréhensibilité d'un schéma. Par exemple, si l'on a trois schémas exportés traitant respectivement de client-particulier, de client-société, de client-indépendant passant des commandes de produits (exemple d'entreprise organisée en départements traitant chacun une catégorie de clients), la préférence sera données à une structure introduisant le concept de client, spécialisé en client particulier, client-société, client-indépendant et relié au concept de produit par un type d'association plutôt que définir trois types d'associations pour chaque client. A cela, on peut encore préférer le rôle multi-domaines(Cfr section 4.2.5)

Cependant, parfois il est impossible impossible de respecter ces trois exigences orthogonales dans leur totalité. C'est pourquoi des compromis devront être faits: par exemple, il faudra notamment veiller à ne pas trop minimiser le schéma, ce qui pourrait compromettre la compréhensibilité de ce dernier, voire sa complétude.

D. Les mappings

Les mappings constituent un mécanisme interne permettant de définir les correspondances entre le schéma intégré et les schémas exportés. Leur rôle est essentiel étant donné l'absence de données réelles au niveau du schéma intégré. En émettant l'hypothèse que les données se situent au niveau des schémas exportés - pour faciliter l'exposé du problème -, ces règles de mappings doivent permettre de construire les occurrences -temporaires- du schéma intégré à partir de celles des schémas exportés. Ainsi une occurrence du schéma intégré sera construite au départ d'une ou plusieurs occurrences des schémas exportés.
Puisque ces règles sont internes au système des BDH, elles ne doivent pas nécessairement être compréhensibles des utilisateurs. En revanche, le formalisme utilisé pur définir ces règles devra être assez **puissant** pour pouvoir traiter tous les cas possibles de correspondances. Nous pourrions en effet nous limiter à un mapping homogène (mapping associant uniquement des éléments de même type) mais il suffit d'examiner l'exemple de la figure 4.22 pour se convaincre que les mappings devront être hétérogènes (mapping permettant la correspondance entre des types différents(TE avec TA, TE avec attribut,...) afin de fournir un schéma intégré minimal.
Les règles de mapping devront être précises et exactes en ce sens qu'aucune ambiguïté ne puisse se présenter à la lecture. Une quelconque ambiguïté ou erreur mènerait inévitablement à la production de résultats incorrects. Ainsi, on exigera que chaque occurrence du schéma intégré soit décrite de façon univoque par une règle de mapping précisant les occurrences correspondantes dans les schémas exportés. On peut déjà préciser que deux types de règles de mapping devront être définis:

- mapping définissant comment construire les occurrences du schéma intégré au départ des occurrences des schémas exportés

- mapping définissant pour chaque schéma exporté comment déduire les occurrences des schémas exportés au départ des occurrences du schéma intégré

Nous exposerons ultérieurement un modèle de description de règles de mapping.

4.4.3. Typologie des conflits entre schémas

Un conflit entre schémas se présente quand deux ou plusieurs schémas représentent un même concept du monde réel de différentes façons. Deux causes sont à l'origine de ces conflits:

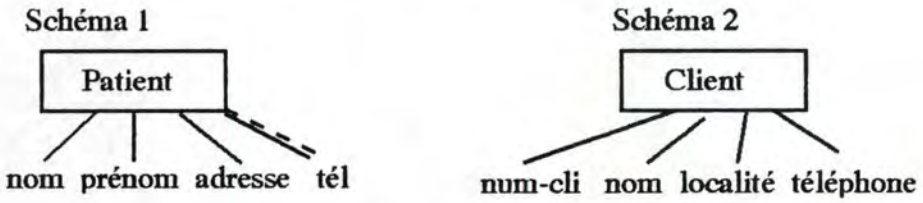
- les schémas des bases de données locales ont été développés de façon indépendante dans des modèles différents et par des personnes différentes. Les concepts du monde réel ont donc été structurés différemment.

- le relativisme sémantique du modèle EA/E nous a permis de définir des schémas composants en rapport avec les structures fournies par les schémas

locaux des BD. Avec l'homogénéisation des schémas, seul le problème d'hétérogénéité syntaxique a été résolu. Pour chaque schéma local, on a simplement opérer un travail de transformation conformément à ce schéma et indépendamment des autres schémas.

Donnons à présent une typologie des conflits que l'on peut rencontrer lorsque l'on compare des schémas représentant le même univers de discours. Nous reprenons la typologie exposée à la section 1.4. et nous l'illustrons par quelques exemples.

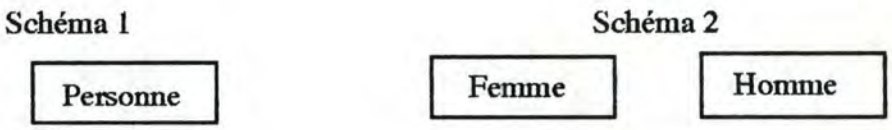
Conflit de description: ce type de conflit est observé quand des classes identiques d'objets sont décrites par des ensembles différents de propriétés. Il s'agit des conflits de noms(homonymie et synonymie), les différences d' échelle, les différences de domaines, de cardinalités, de contraintes...



4.23.: Exemple de conflit de description

Ces deux schémas représentent les personnes qui sont clientes chez un médecin: le schéma 1 les désigne comme des patients et le schéma 2 comme des clients et tout deux leurs reconnaissent des attributs différents

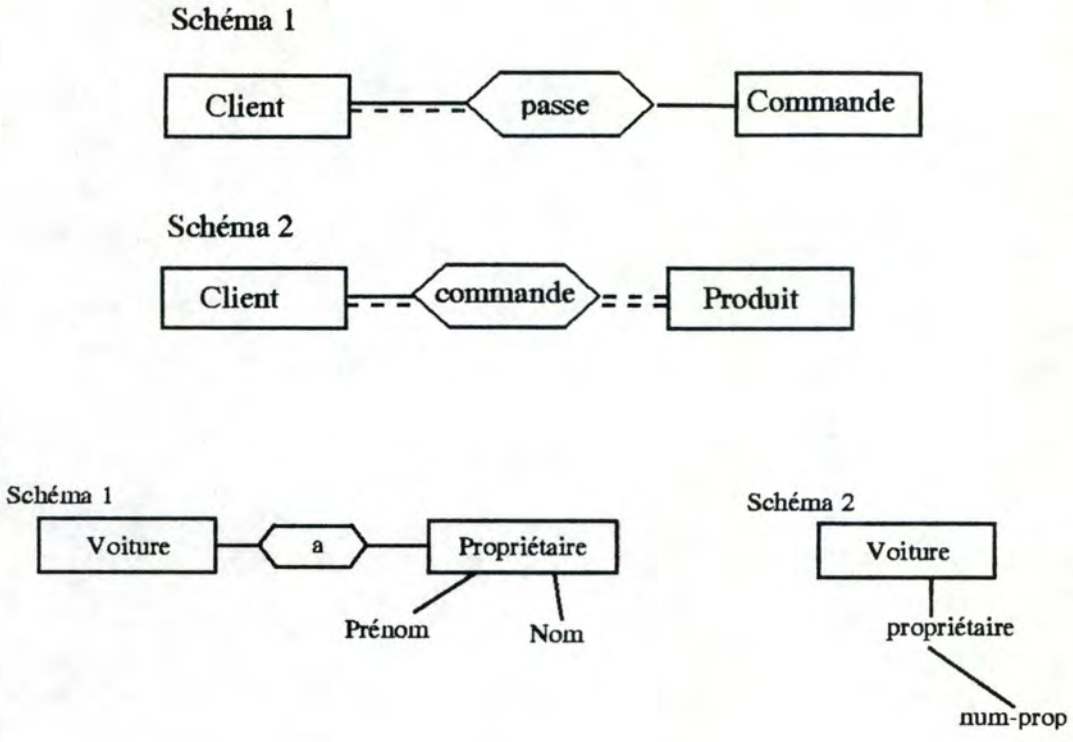
Conflit sémantique: ce type de conflit est observé quand deux schémas adoptent des classifications différentes pour les mêmes ensembles d'objets. Cela débouche généralement sur des ensembles d'objets qui se recouvrent.



4.24.: Exemple de conflit sémantique

Les deux schémas ont pour univers de discours l'ensemble des individus d'une ville. Le schéma 1 considère uniquement des personnes tandis que le schéma 2 distingue les hommes des femmes.

Conflit structurel: ce type de conflit est observé lorsque deux sous-ensembles d'objets identiques du monde réel sont représentés par des structures différentes.



4.25: Exemples de conflit de structure

Ces trois types de conflits peuvent être cumulés de sorte que les relations existantes entre les différents schémas soient difficiles à distinguer.

L' exemple de la figure 4.26. illustre ce problème dont voici les conflits:

Conflit de description: attributs de client et de client-indépendants

Conflit de structure: le schéma 1 représente le concept de commande par un TE tandis que que le schéma 2 utilise une structure d'association entre le TE client et produit pour représenter le même concept de commande.

Conflit sémantique: le schéma 1 reprend tous les clients tandis que le schéma 2 ne considère qu'un sous-ensemble de clients, les clients-indépendants

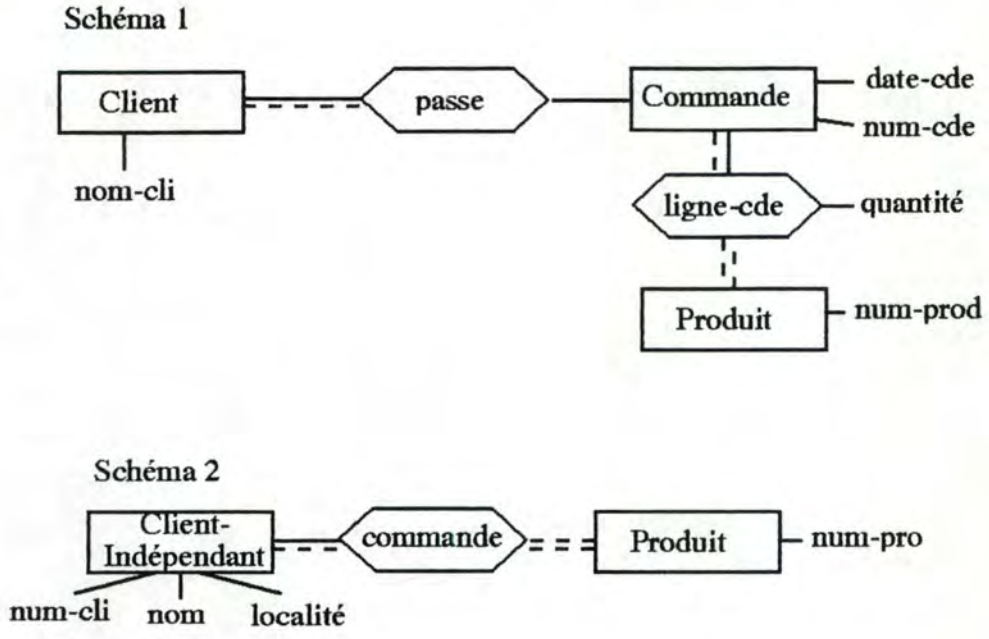


Figure 4.26.: Illustration des conflits existants entre schémas

4.4.4. Méthodes d'intégration de schémas

Il existe de nombreuses méthodes d'intégration qui se différencient selon le modèle utilisé, les types de conflits pris en considération, les étapes à suivre pour aboutir au schéma intégré; mais elles peuvent se regrouper en deux catégories en fonction de l'approche qu'elles utilisent: l'**approche procédurale** et l'**approche déclarative**. La première approche propose un *langage de restructuration/intégration* et la seconde un *modèle de définition de correspondances.*
Chacune des méthodes qu'elles soient de type procédurale ou déclarative peut être de type **manuel, semi-automatique** ou **automatique**. Elles diffèrent uniquement par le degré d'implication du responsable d'intégration: les méthodes manuelles imposent à l'utilisateur de tout faire, les semi-manuelles de réaliser une partie du travail d'intégration et les automatiques lui imposent un travail de contrôle.

Avant d'expliquer ces deux approches, montrons combien il est impossible à l'heure actuelle de définir une méthode d'intégration de schémas automatique. Cela nécessiterait que toute la sémantique de chaque schéma soit complètement spécifiée de telle sorte qu'aucune ambiguïté d'interprétation ne subsiste. Cela s'avère impossible pour les raisons suivantes: aucun modèle sémantique disponible aujourd'hui n'est capable de représenter complètement un univers de discours; il serait également nécessaire que le schéma contienne beaucoup plus d'information qu'il n'en contient aujourd'hui; de plus, il peut y avoir plusieurs interprétations d'un même univers de discours et celles-ci peuvent varier avec le temps. Etant donné l'impossibilité actuelle de définir une telle méthode, les méthodes semi-automatiques sont parfois qualifiées d'automatiques.

A. Approche procédurale

Ce type d'approche consiste à construire le schéma intégré à l'aide d'un langage de manipulation-restructuration. Il offre des opérations permettant de définir le schéma intégré sur base des schémas exportés. Comme référence, on peut citer [MOTRO 87] et [DAYAL 84]. Ces deux méthodes sont qualifiées de **manuelles** étant donné le degré d'implication de l'utilisateur. Une méthode **semi-automatique** proposerait au responsable d'intégration des opérations de restructuration/intégration qu'il n'aurait plus qu'à valider s'il est d'accord.

[MOTRO 87] a défini une méthode(figure 4.27) qui consiste à regrouper en un seul schéma l'ensemble des schémas à intégrer. Ensuite, à l'aide des opérations du langage que Motro a défini, l'administrateur de la base de données va restructurer ce schéma, c'est-à-dire fusionner des éléments, établir des liens entre d'autres éléments, renommer, créer ou supprimer des éléments,... A chaque opérateur de restructuration, Motro a associé une règle de mapping. Ainsi, à chaque fois que l'on applique un opérateur au schéma, le système crée une règle de mapping permettant d'établir les correspondances entre le schéma intégré et les schémas de départ. Ces règles de mapping sont établies de façon progressive, ce qui signifie qu'une règle de mapping peut être modifiée plusieurs fois et ne sera définitive que lorsque que l'on décidera d'arrêter de restructurer le schéma.Motro a défini 10 opérateurs d'intégration-restructuration en utilisant le modèle fonctionnel comme base à leur définition. L'application de ces opérateurs ne peut se faire que si certaines préconditions portant sur les arguments(les éléments du schéma) des opérateurs sont respectées.

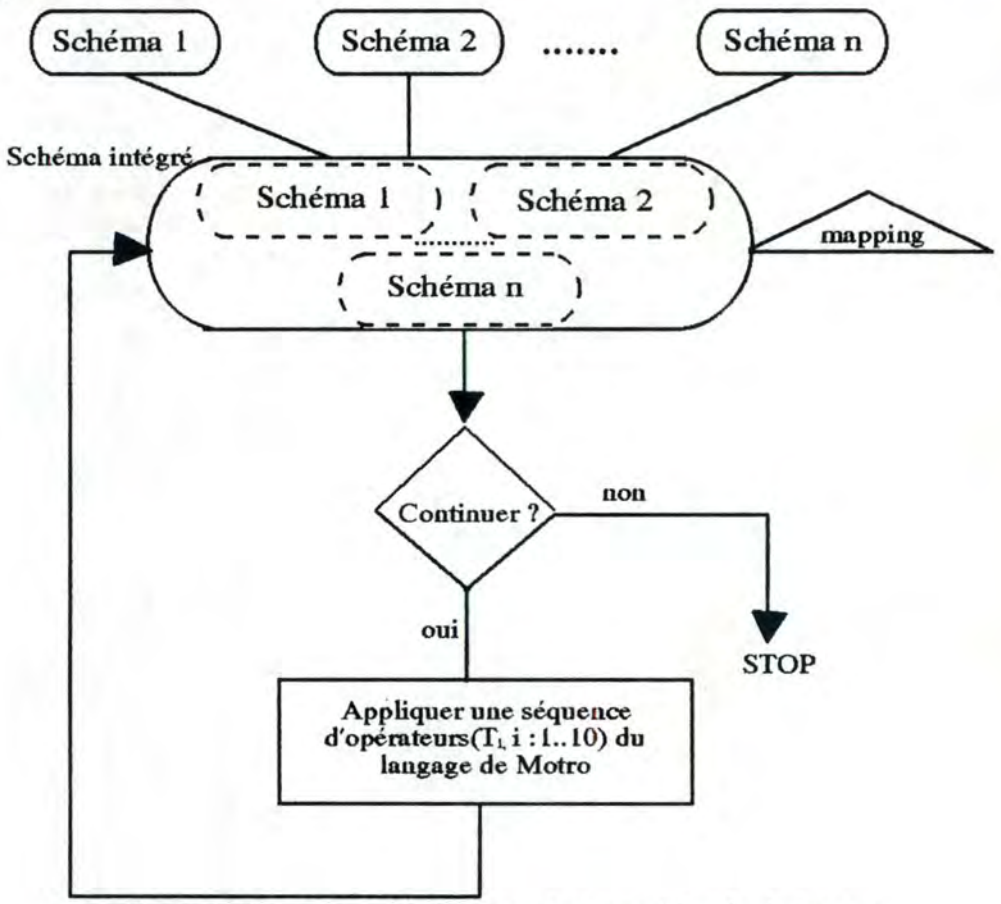


Figure 4.27.: Approche d'intégration procédurale de Motro

Ainsi, le déroulement de la phase d'intégration est entièrement guidé par la personne responsable de la conception du schéma intégré. Cette personne est chargée d'identifier et de résoudre les conflits entre les différents schémas. Elle décide du moment de terminaison de la phase d'intégration. De ce fait, il lui incombe de vérifier la qualité du schéma final en rapport avec les caractéristiques que nous avons déterminées précédemment.
La décision de continuer le processus d'intégration sera évaluée en se posant la question suivante: le schéma intégré est -il minimal et compréhensible?

Décrivons brièvement les fonctions des opérateurs définis par Motro sur base du modèle fonctionnel. Une présentation plus complète, reprenant les préconditions et postconditions des opérateurs peut être trouvée dans [MOTRO 87].

Rename : change le nom d'une classe d'éléments
Aggregate : crée une classe intermédiaire
Telescope : ôte une classe intermédiaire
Fold : fait absorber par une classe générique l'une des classes spécifiques
Join : crée une spécialisation de deux classes
Meet : crée une généralisation de deux classes
Combine : fusionne deux classes de types identiques
Connect : établit un lien de généralisation entre deux classes
Add : rajoute dans une classe un attribut ayant une valeur constante
Delete : permet d'enlever des attributs d'une classe

[DAYAL 84](figure 4.28) ayant abordé le problème par cette approche, se différencie de Motro par le fait qu'il utilise un langage d'intégration semblable au langage de manipulation de données QUEL. Il s'agit ici plus d'un langage de définition de schéma plutôt qu'un langage de restructuration. En effet, on définit le schéma intégré de façon progressive en y incluant des entités des différents schémas initiaux, on y crée de nouvelles entités, des relations entre entités(spécialisation-généralisation), on y renomme des entités et on établit des règles de correspondances entre domaines.

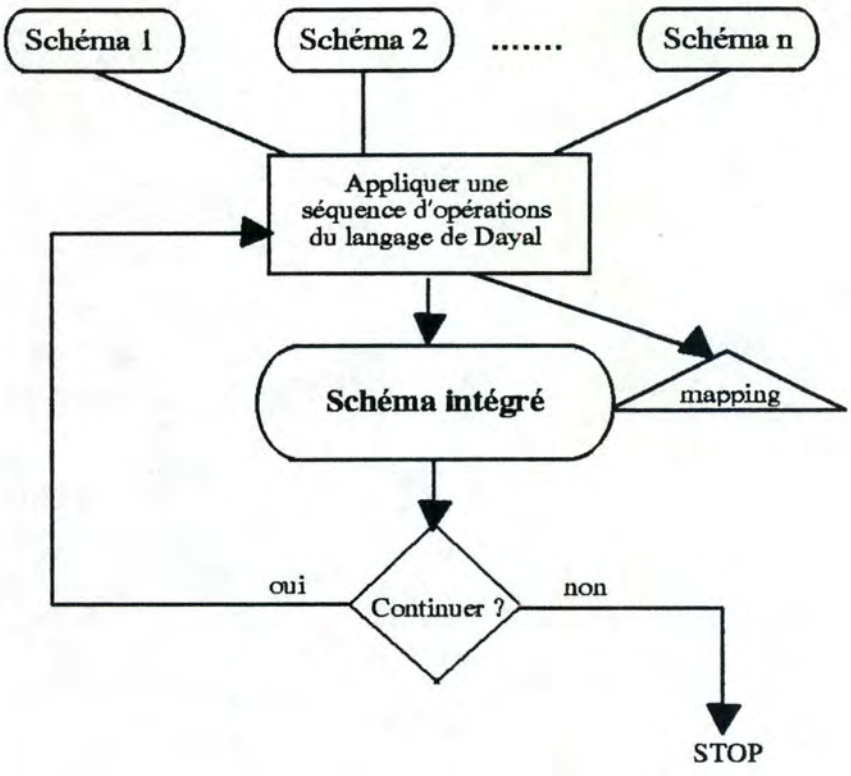


Figure 4.28. :Méthode d'intégration procédurale de Dayal

La décision de continuer le processus d'intégration dépendra des réponse apportées aux deux questions suivantes:

1. Est-ce que tous les éléments des schémas exportés ont été repris dans le schéma intégré?
2. Le schéma intégré est-il minimal et compréhensible?

Une réponse positive aux deux questions signalera la fin de la phase d'intégration.

B. Approche déclarative

Cette approche se distingue de la précédente par le fait qu'elle utilise un langage assertionnel permettant de décrire les correspondances entre schémas à intégrer. Ainsi, nous pouvons distingue dans cette approche deux étapes, qui se déroulent de façon séquentielle :

1. ***Etape de définition des correspondances inter - schémas*** : étape manuelle ou semi-automatique suivant que l'utilisateur est seul pour définir les correspondances ou s'il est aidé par le système d'intégration.

2. ***Etape d'intégration*** proprement dit, basée sur les correspondances définies au préalable : étape réalisée automatiquement mais sous le contrôle du responsable d'intégration.

Une **correspondance** entre deux schémas peut se définir comme une assertion spécifiant la **relation** existant entre un élément du premier schéma et un ou plusieurs éléments du second schéma. De façon plus générale, un correspondance inter-schémas est un ensemble de correspondances entre schémas pris deux à deux. Chaque correspondance appartient à un **type de correspondance**. Plusieurs typologies de correspondances ont été proposées par divers auteurs mais nous n'exposerons que celle qui a été le plus largement reconnue dans la littérature. A chaque type de correspondance est associé une **règle d'intégration**. Elle permettra de fournir le résultat intégré des éléments participants à la correspondance.

Il existe deux catégories de correspondances, les correspondances **homogènes** et les correspondances **hétérogènes**. Comme nous allons le voir, cette distinction mène à des méthodes d'intégration de puissances différentes.

B.1. Méthode basée sur les correspondances homogènes

Cette catégorie de correspondance se caractérise par le fait qu'elle ne permet que l'expression de relations entre concepts représentés par des structures identiques. Dans le cas du modèle entité-association EA/E, on est donc limité à définir des correspondances inter-TE, inter-TA ou inter-attributs. Or on a montré au moyen des exemples de la figure 4.25 que nombreux sont les cas où se présentent des ***conflits de structure*** (exemple d'un TE d'un premier schéma en correspondance avec un TA d'un second schéma,...). De tels conflits ne peuvent faire l'objet d'une correspondance dans ce modèle de correspondances homogènes. Une alternative se présente face à un tel cas de figure:

- **soit** on se contente des correspondances qu'il nous est possible d'exprimer à l'aide du modèle proposé et on lance l'étape d'intégration sur base de ces correspondances. Exception faite des situations d'intégration ne présentant pas de conflits structurels, le schéma intégré en résultant sera complet mais non minimal et peut être peu compréhensible vu la redondance de concepts identiques ou similaires

- **soit** on procède à une étape préliminaire, généralement appelée ***étape de "conforming"***, qui consiste à ramener tous les concepts en correspondance à un même niveau de perception au sein des divers schémas. Par exemple, si l'on est face à un conflit TE-Attribut, on transformera l'attribut en question par un TE. Ainsi on élimine tout conflit de structure. Cette opération sera réalisée ***soit*** entièrement manuellement par le responsable de l'intégration, ***soit*** de façon semi-automatique, c'est-à-dire en utilisant des transformations de schéma définies sur le modèle EA/E. Ces transformations devront respecter la sémantique initiale (cfr. 4.2) et fournir les règles de mapping pour pouvoir retrouver les données sous-jacentes. On peut citer comme exemple, la transformation d'un attribut en type d'entités, d'un type

d'association en un type d'entités.... Dans [HAINAUT 91a] plusieurs transformations de schémas de type entité-association sont proposées. La figure 4.29 illustre une de ces transformations.

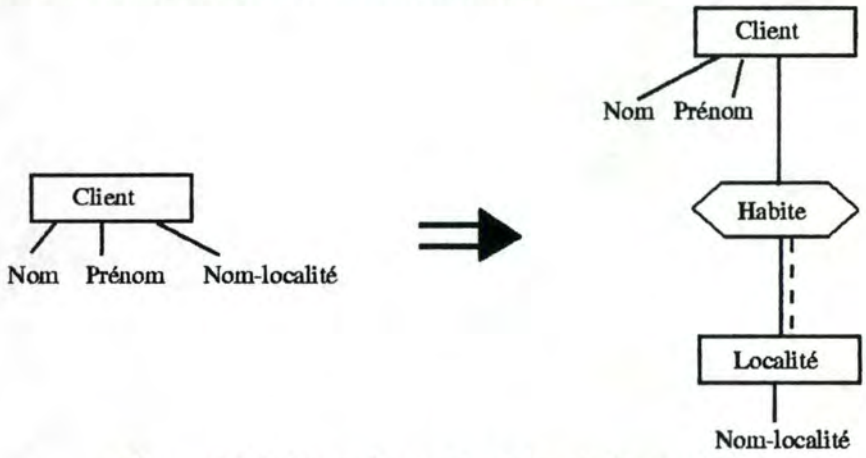


Figure 4.29 : Transformation d'un attribut en TE

Ainsi on a l'avantage de pouvoir faire face à tous les conflits de structures et de disposer d'un modèle d'expression de correspondances simple. Par contre l'inconvénient réside dans l'étape préliminaire qui peut s'avérer coûteuse en temps bien que "one-shot" et complexe à réaliser.

Une des **typologies** de correspondances que l'on rencontre dans ces méthodes est la suivante. Les correspondances entre TE et celles entre TA sont basées sur le fait que les deux éléments considérés représentent le même sous-ensemble du monde réel, désigné par l'expression *Real World State(RWS, Etat du Monde Réel)*. La figure 4.30 illustre la relation entre schéma, occurrences et RWS. On y constate la relation entre les TE "propriétaire" des schémas 1 et 2 et cela malgré des description distinctes. C'est pourquoi on se base sur le concept de RWS pour définir les correspondances entre les éléments. Par contre les correspondances entre attributs sont basées sur l'équivalence de leurs domaines.

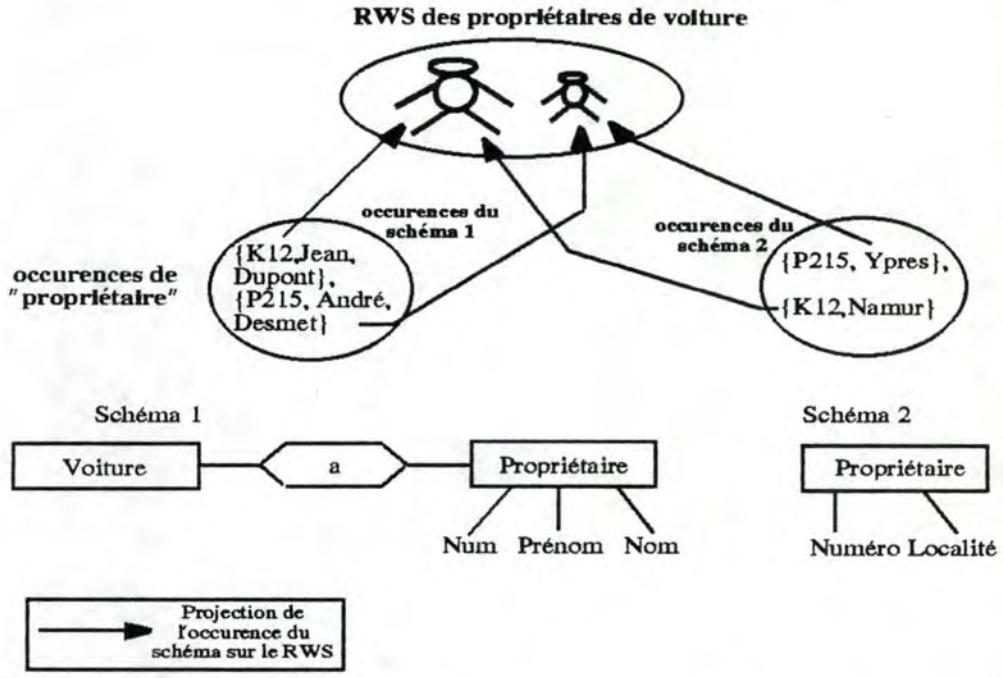


Figure 4.30 : Relation entre schémas, occurrences, Real World State

Définissons à présent les concepts de RWS d'un TE et d'un TA.

RWS d'un TE

"Soit E un TE; l'état du monde réel d'un TE E, noté RWS(E), est l'ensemble des objets du monde réel représenté par les occurrences de E. Il y a une correspondance 1-1 entre la population de E et le RWS(E)" [SPACCAPIETRA 90].

RWS d'un TA

"Soit R un type de relation, liant les types d'entités $E_1,E_2,...,E_n$; l'état du monde réel de R, noté RWS(R), est l'amas(ensemble pouvant contenir des doubles) des tuples d'objets du monde réel [< $o_1, o_2,...,o_n$ >] tel que $\forall i$, $o_i \in RWS(E_i)$ et tel que les objets dans un tuple sont liés dans le monde réel par l'association représentée par une occurrence de R.
NB: [...] : représente un amas du fait qu'un même ensemble d'entités peut participer à plusieurs occurrences d'un même TA.
Il y a une correspondance 1-1 entre la population de R et le RWS(R)"[SPACCAPIETRA 90].
La typologie de correspondance que l'on va exposer est basée sur les quatre relations fondamentales que l'on peut définir entre deux ensembles: l'équivalence, l'inclusion, l'intersection et la disjonction(ou exclusion).

Etant donné deux éléments E1 et E2 de mêmes structures(TE ou TA), on peut définir les correspondances suivantes:

Equivalence : E1 ≡ E2

L'assertion que E1 est équivalent à E2 est vraie ssi à tout instant:

RWS(E1) = RWS(E2)

Ex: Deux départements distincts d'une entreprise disposent d'un fichier reprenant les produits proposés par l'entreprise. Les produits sont les mêmes pour les deux départements, c'est pourquoi on définira une assertion d'équivalence.

Inclusion : E1 C E2

L'assertion que E1 est inclus dans E2 est vraie ssi à tout instant:

RWS(E1) C RWS(E2)

Ex: Chaque département d'une entreprise dispose de son propre fichier clients. Chaque département traite les clients d'une région particulière. Il existe en outre un fichier global de tous les clients toutes régions confondues dans le département central. Ainsi nous dirons que les fichiers clients de chaque département sont inclus dans le fichier global; cela nous amènera à définir des assertions de correspondances d'inclusion.

Intersection : E1 ∩ E2

L'assertion que E1 et E2 se recouvrent est vraie ssi à tout instant **il est possible** que

RWS(E1) ∩RWS(E2) ≠ ∅

Ex: Chaque département dispose d'un fichier de fournisseurs. Etant donné qu'un même fournisseur peut servir plusieurs départements , on retrouvera dans les fichiers de chaque départements des fournisseurs communs. C'est pourquoi on établira des correspondances d'intersection.
A certains moments il peut ne pas exister de fournisseur commun à plusieurs départements. Cela explique la définition de la correspondance en des termes potentiels ("il est possible que").

Disjonction : E1 ≠ E2

L'assertion que E1 et E2 sont disjoints est vraie ssi à tout instant:

RWS(E1) ∩ RWS(E2) = ∅

Ce type de correspondance permet d'éviter de déduire des correspondances erronées par déduction d'autres correspondances ou d'affiner des correspondances. Par exemple, les correspondances E1 C E3 et E2 C E3 seraient complétées par E1 ≠ E2 pour éviter de déduire de fausses relations entre E1 et E2.

Ex: Chaque département dispose d'un fichier de ses employés. Comme un employé est assigné à un et un seul département, on définira des correspondances de disjonction entre ceux-ci.

Les relations entre attributs ne sont généralement définies que s'il existe des correspondances entre les éléments auxquels ils appartiennent. Il est évident qu'un attribut n'a d'existence que s'il est attaché à un TE ou un TA. On pourra également définir quatre types de correspondances entre attributs pour autant que les correspondances entre domaines soient établies par le responsable de l'intégration. Nous exposerons plus amplement les correspondances possibles entre attributs dans le paragraphe B.2 de cette section.

La figure 4.31 fournit une synthèse graphique de la méthode déclarative basée sur des correspondances homogènes.

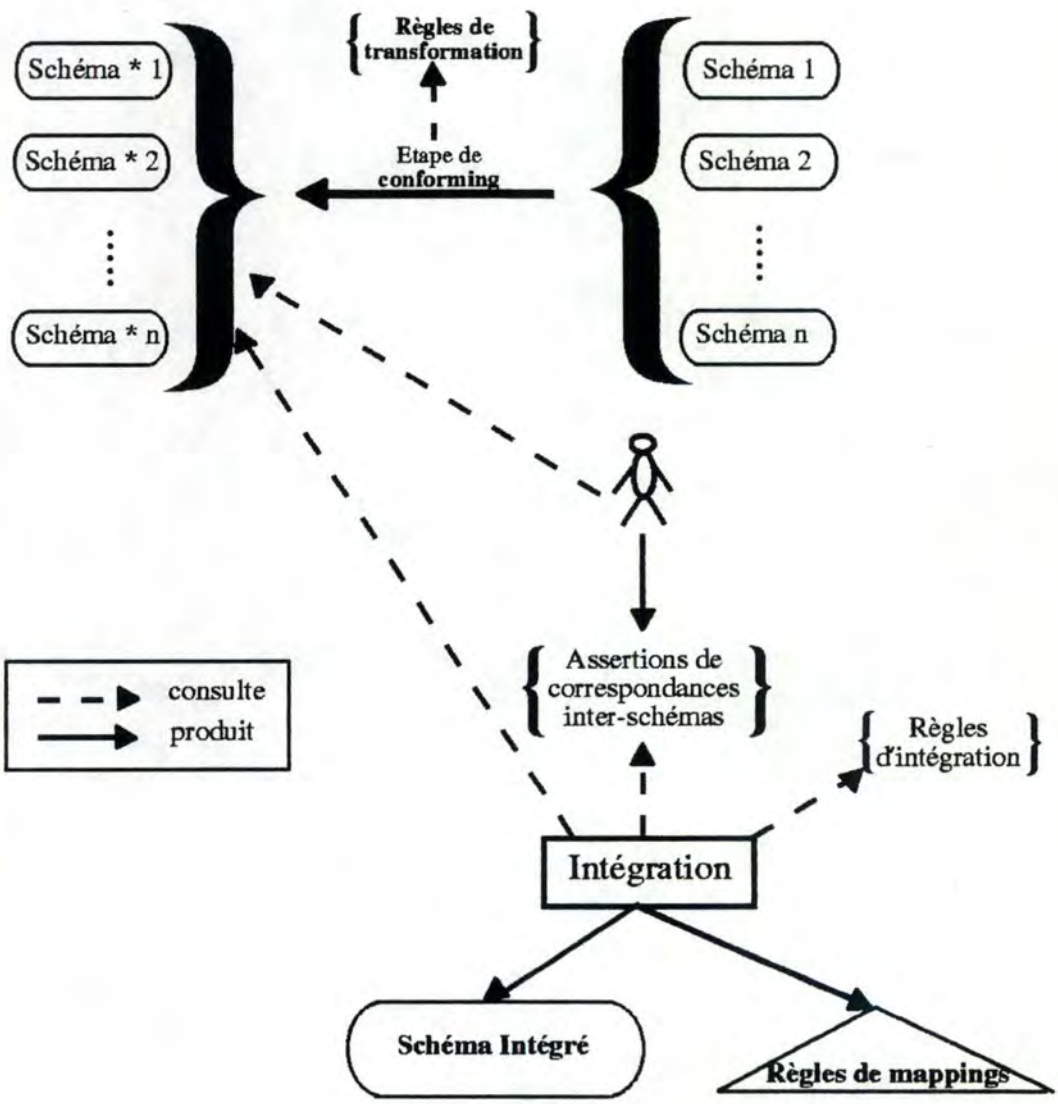


Figure 4.31.: Approche déclarative avec correspondances homogènes

Trois étapes ont ainsi été définies: l'étape de "conforming", l'étape de définition des correspondances, l'étape d'intégration proprement dite qui fournit le schéma intégré et génère les règles de mapping entre le schéma intégré et les schémas exportés.

Exemple

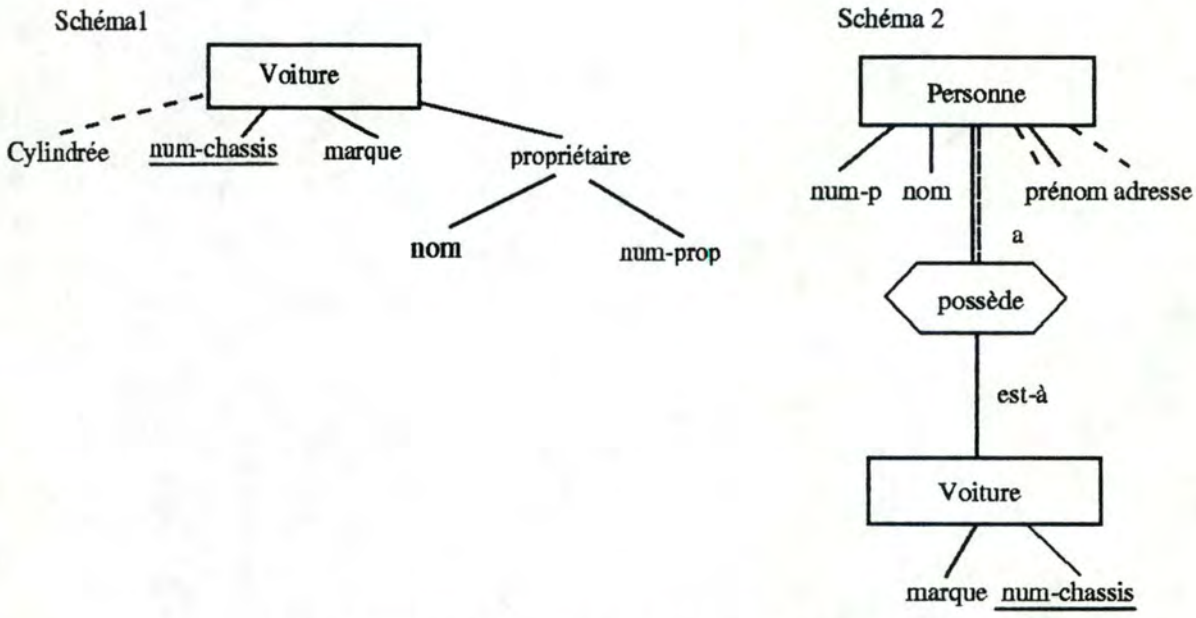


Figure 4.32.: Exemple de schémas à intégrer

Sans passer par l'étape de conforming, les correspondances seraient les suivantes :

Schéma 2.voiture ≡ Schéma 1. voiture (marque= marque, num-chassis = num-chassis)

Elle peut se traduire comme suit: les TE voitures de chacun des schémas sont équivalents et les attributs marque et num-chassis des deux schémas sont égaux.

Sur base de celles-ci, le schéma intégré serait le suivant:

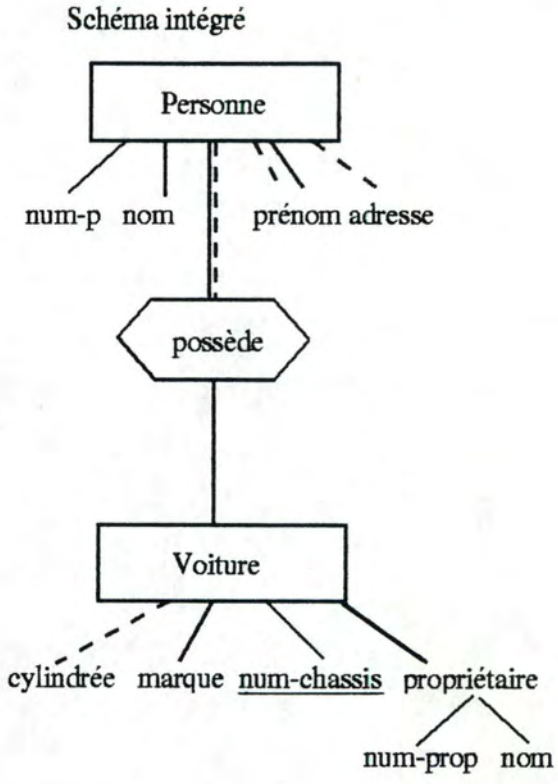


Figure 4.33 : Mauvais schéma intégré(non minimal)

La redondance paraît évidente concernant le concept de "propriétaire".
Appliquons au schéma 1 la phase de "conforming"; cette dernière nous fournit le schéma de la figure 4.34 à partir duquel on établit les correspondances suivantes :

Schéma 1*.Propriétaire(num-prop, nom) ≡ Schéma 2.Personne(num-p,nom)
Schéma 1*.Voiture(marque, num-chassis) ≡ Schéma 2.Produit(marque, num chassis)
Schéma 1*. appartenance ≡ Schéma 1.possède

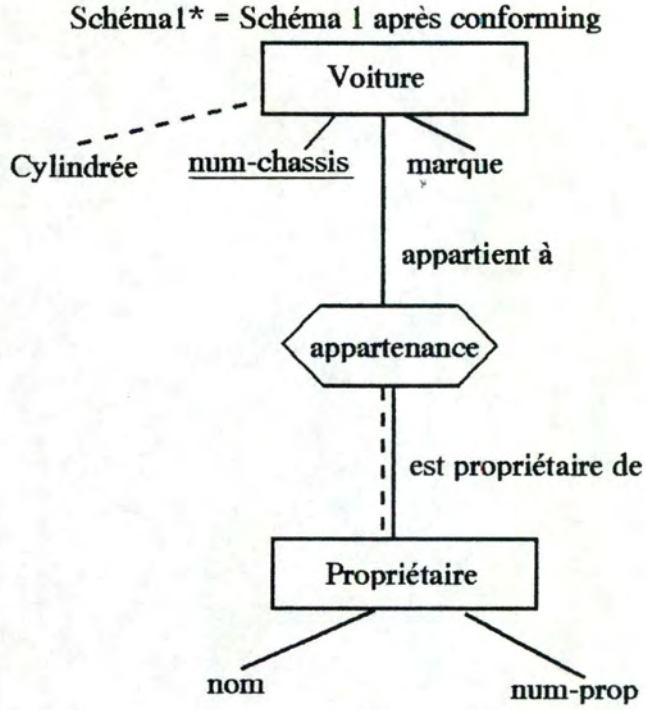


Figure 4.34.: Schéma 1 après phase de conforming

Le schéma intégré serait dès lors celui de la figure 4.35.

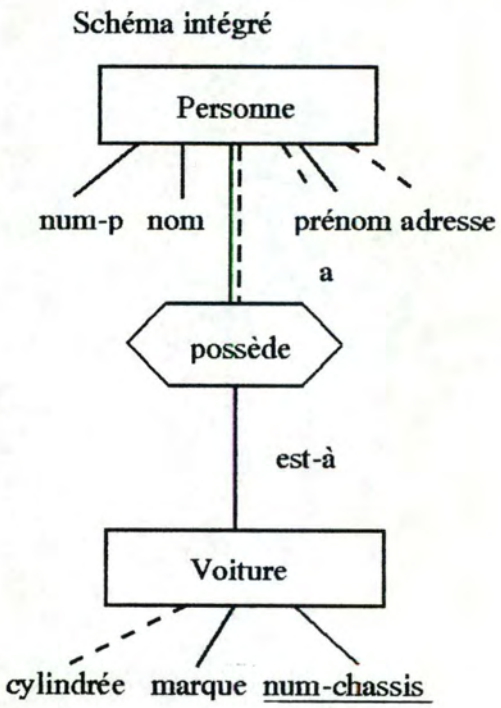


Figure 4.35 : Bon schéma intégré

B.2. Méthode basée sur les correspondances hétérogènes

A l'opposé des correspondances homogènes, on trouve les correspondances hétérogènes, celles qui autorisent la définition de relations entre des éléments de structures différentes(TA, TE, attribut). Le problème se situe au niveau de la définition du modèle de description des correspondances ainsi que du modèle de définition des règles de mapping qui doit être capable d'exprimer des relations entre occurrences de structures différentes. Ce problème est assez complexe, c'est pourquoi les méthodes basées sur cette catégorie de correspondances sont peu fréquentes. Nous citerons la méthode semi-automatique de [SPACCAPIETRA 90,91].
L'origine de la puissance de ce modèle d'expression de correspondances réside dans le principe du "conforming implicite". En effet, la structure des correspondances indique la présence d'un conflit structurel. Ainsi, il reste à procéder à une étape de conforming automatique et interne (le responsable de l'intégration n'a pas à s'en préoccuper). Toutefois les règles de mapping seront définies en conformité avec les schémas initiaux et non avec les schémas ayant subi le phase de conforming.
Pour établir de telles correspondances, il faut établir une base commune de comparaison à tous les concepts de structuration du modèle(TE, TA, attribut). Nous avons vu dans les méthodes à correspondances homogènes que l'on comparait différemment les éléments suivant qu'ils étaient des attributs ou bien des TE ou TA. Dans le cas présent, il faudra une base commune que nous définirons en étendant le concept de RWS à la structure d'attribut.

RWS d'un attribut

"Soit A un attribut, l'état du monde réel de A, noté RWS(A), est l'ensemble des objets du monde réel représentés par les valeurs de A dans la base de données.
Il y a une fonction surjective totale des valeurs de A vers RWS(A). En effet, la même valeur d'attribut peut être prise plusieurs fois." [SPACCAPIETRA 90].

Comme on pourra avoir à comparer des parcours qui mettent en liaison les éléments d'un même schéma, on introduit la notion de chemin: "soit X_1, X_2,..,X_n des noeuds dans un schéma (TE, TA et/ou attributs) tel que $\forall$ i $\in$ {1,2, n-1}, X_i est lié à X_{i+1}, soit par un lien de parenté TE-attribut, soit par un lien de rôle(TE, TA): X_1-X_2-...-X_n est appelé un chemin" [SPACCAPIETRA 90].

RWS d'un chemin

"L'état du monde réel d'un chemin X_1-X_2-....-X_n, RWS(X_1-X_2-...-X_n), est l'ensemble des paires d'objets du monde réel [<o_1,o_n>], telles que $o_1 \in$ RWS(X_1) et $o_n \in$ RWS(X_n), et il existe des objets o_2,o_3,...,o_{n-1} tels que $\forall$ i $\in$ {1,2,..,n-1}, $o_i \in$ RWS(Xi), avec o_i et o_{i+1} liés par l'association du monde réel représentée par le lien X_i - X_{i+1}." [SPACCAPIETRA 90].

Typologie des correspondances

Cette typologie constitue une adaptation de celle proposée au paragraphe B.1 en tenant compte de correspondances hétérogènes.

"Soit E_1, E_2 deux éléments de type quelconque(TE, TA, attribut), E_1 appartient au schéma 1 et E_2 au schéma 2; o_1 et o_2 sont deux objets du monde réel tels que $o_1 \in RWS(E_1)$ et $o_2 \in RWS(E_2)$

Equivalence : $E_1 \equiv E_2$

L'assertion que E_1 est équivalent à E_2 est vraie ssi à tout instant:

1. *$RWS(E_1) = RWS(E_2)$*
2. *Il y a une fonction totale bijective $f : RWS(E_1) \rightarrow RWS(E_2)$, telle que $o_2 = f(o_1)$ ssi o_1 et o_2 ont la même sémantique*

Inclusion : E_1 C E_2

L'assertion que E_1 est inclus dans E_2 est vraie ssi à tout instant:

1. *$RWS(E_1)$ C $RWS(E_2)$*
2. *Il y a une fonction totale injective $f : RWS(E_1) \rightarrow RWS(E_2)$, telle que $o_2 = f(o_1)$ ssi o_1 et o_2 ont la même sémantique*

Intersection : $E_1 \cap E_2$

L'assertion que E_1 et E_2 se recouvrent est vraie ssi ***à tout instant:***

1. ***Il est possible que*** *$RWS(E_1) \cap RWS(E_2) \neq \emptyset$*
2. *Il y a une fonction injective partielle $f : RWS(E_1) \rightarrow RWS(E_2)$, telle que $o_2 = f(o_1)$ ssi o_1 et o_2 ont la même sémantique*

Disjonction : $E_1 \neq E_2$

L'assertion que E_1 et E_2 sont disjoints est vraie ssi à tout instant:

1. *$RWS(E_1) \cap RWS(E_2) = \emptyset$*
2. *Il n'y a pas de fonction $f : RWS(E_1) \rightarrow RWS(E_2)$, telle que $o_2 = f(o_1)$ ssi o_1 et o_2 ont la même sémantique " [SPACCAPIETRA 90]*

La première partie de l'assertion de correspondance (1) exprime le cas des correspondances homogènes tandis que la seconde partie(2) traite les correspondances hétérogènes.

Ces quatre types de correspondances constituent la base du modèle de description des correspondances. Cette typologie se limite à des correspondances binaires c'est-à-dire ne faisant intervenir que deux éléments de deux schémas distincts.
On complétera ces assertions de correspondances par une clause supplémentaire appelée *"With Corresponding Attributes(WCA)"* et qui spécifie les relations entre les attributs des éléments en

correspondance. Notons que la première correspondance entre les attributs d'éléments en correspondance portent sur les identifiants des éléments pour permettre les relations entre les occurences des schémas.

Le **formalisme de description** des correspondances sera donc le suivant:

Soit ERA_i, un TE ou un TA ou un attribut.
Soit ERA_1 <corr> ERA_2 une assertion de correspondance avec $< o_1,o_2 >$ une paire d'objets en correspondance, $o_1 \in RWS(ERA_1)$ et $o_2 \in RWS(ERA_2)$.

Soit e_1, e_2 les occurences représentant o_1 et o_2 . Soit $A_1,A_2,...,A_n$ des attributs de ERA_1 et B_1, $B_2,...,B_m$ des attributs de ERA_2(si ERA est un attribut simple il est implicitement considéré comme ayant lui-même comme composant unique) **alors**

ERA_1 <corr> ERA_2 **With Corresponding Attributes** : $attcor_1(A_1,B_1)$, $attcor_2(A_2,B_2)$,...,
La première correspondance entre attribut $attcor_1(A_1,B_1)$concerne les identifiants des deux élements concernés et ce afin de pouvoir ultérieurement réaliser l'intégration des données.
$attcor_i(A_i,B_i)$ est une assertion de correspondance qui est vraie ssi ERA_1 <corr> ERA_2 est vraie et pour chaque $attcor_i(A_i,B_i)$:

si $attcor_i(A_i,B_i)$ est $A_i = B_i$ **alors** pour chaque paire o_1,o_2 : $e_1.A_i = o_2.B_i$

si $attcor_i(A_i,B_i)$ est $B_i = f_i(A_i)$ **alors** f_i est une fonction surjective(définie par le responsable d'intégration) définie de Valeur(Ai) sur valeur(Bi) telle que à tout instant pour chaque paire o_1,o_2 : $e_1.B_i = f_i(o_2.A_i)$. Valeur (A) avec A attribut, représente l'ensemble des valeurs de A dans toutes les occurrences existantes de l'élément paire de A; il s'agit dès lors d'un sous-ensemble du domaine de A. La fonction f_i peut être une fonction de sélection(sous-ensemble de), de transformation(FB-FF), de comptage,...

si $attcor_i(A_i,B_i)$ est $A_i \cap B_i$ **alors** il est possible que pour quelques paires o_1,o_2 : $e_1.A_i \cap o_2.B_i \neq \emptyset$; pour un attribut simple cela devient $e_1.A_i = o_2.B_i$

si $attcor_i(A_i,B_i)$ est $A_i \neq B_i$ **alors** pour chaque paire o_1,o_2 : $e_1.A_i \cap o_2.B_i = \emptyset$ mais il est possible de définir un domaine D tel que $D \supset$ Valeur(A_i) et $D \supset$ Valeur(B_i).

Exemple

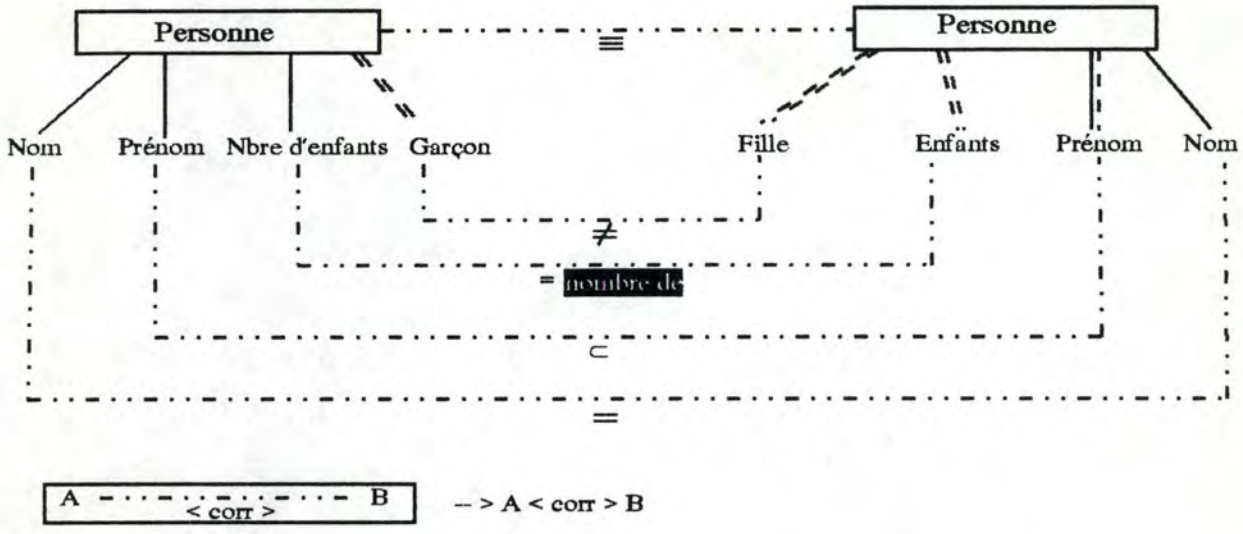


Figure 4.36 : Exemple de correspondances

La fonction "nombre de" présentée à figure 4.36 est définie par l'opérateur algébrique "Card", défini en Annexe 1. Cet opérateur, appliqué à un attribut, donne le nombre de valeurs contenues dans cet attribut.

Etant donné que l'on admet des correspondances hétérogènes, on se doit d'étendre le modèle des correspondances par la définition de correspondances entre chemins. L'exemple de la figure 4.37 illustre cette nécessité.

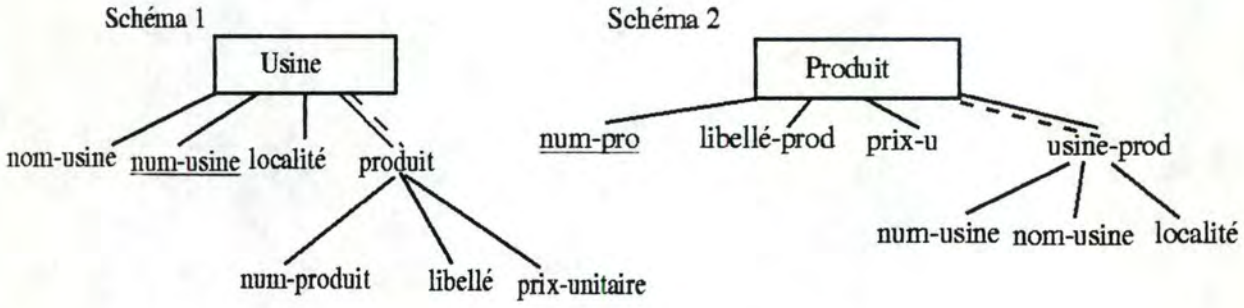


Figure 4.37.: Schémas à intégrer

Sans spécifier les correspondances entre chemins, on définira les correspondances suivantes:

Schéma 1. Usine.produit ≡ Schéma 2. Produit
WCA num-produit= num-pro, libellé = libellé-prod, prix-unitaire=prix-u

Schéma 1.Usine ≡ Schéma 2. Produit.Usine-prod

WCA num-usine = num-usine, nom-usine = nom-usine, localité= localité

Sur base de ces correspondances, le schéma serait le suivant:

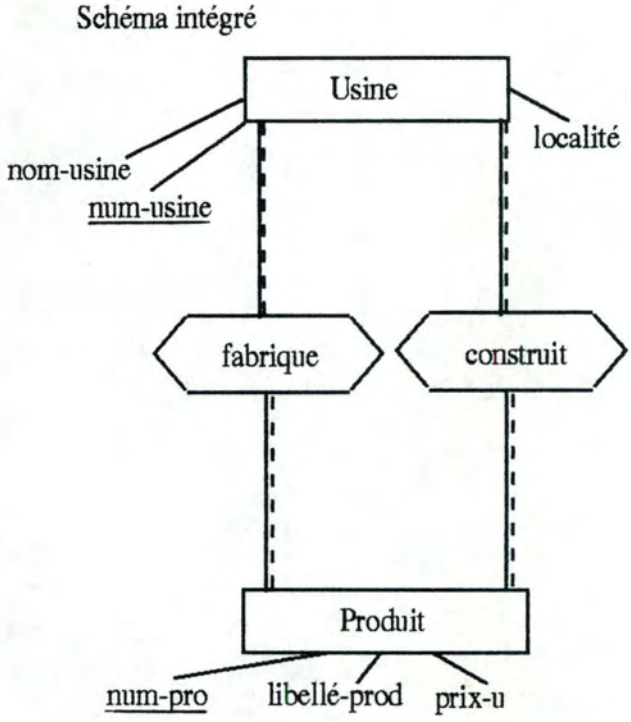


Figure 4.38. : Schéma intégré de la figure 4.37

Ce schéma présente de la redondance au niveau des TA entre ces deux TE. Les associations "fabrique" peuvent se déduire des associations "construit" et réciproquement.

Si l'on définit la correspondance suivante sur base des schémas initiaux :

Schéma 1.Usine.produit - Usine ≡ Schéma 2.Produit - Produit.usine

On obtiendra dès lors un schéma intégré complet, minimal et compréhensible, celui de la figure 4.39.

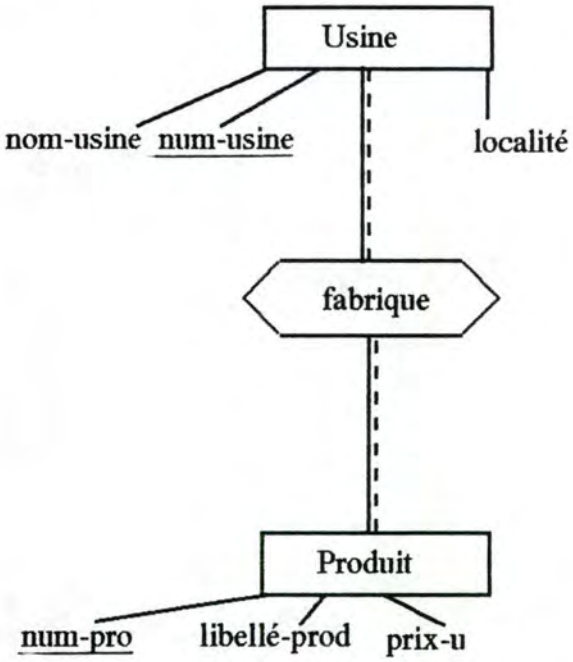


Figure 4.39. : Schéma intégré correct de 4.37

Nous ne pouvions pas déduire cette assertion de correspondance de manière implicite car nous aurions pu avoir deux sémantiques différentes entre une usine et un produit(fabrique, vend) et dans un tel cas les deux TA auraient du être conservés comme à la figure 4.38 car une usine peut ne pas vendre tous les produits qu'elle fabrique ou inversément elle peut vendre des produits qu'elle ne fabrique pas elle-même.

L'assertion de **correspondance entre deux chemins** se définit dès lors comme suit:

"Soit E_1-...-E_n un chemin du schéma 1 et F_1-...-F_p un chemin du schéma 2 tel que E_1 et F_1 sont en correspondance via une fonction f(E_1-->F_1), et E_n et F_p sont en correspondance via une fonction f'(E_n-->F_p).
Soit RWS'(E_1) un sous ensemble de RWS(E_1) défini par restriction aux objets de E_1 impliqués dans la correspondance établie avec F_1; de même pour RWS'(E_n), RWS'(F_1), RWS'(F_p).
Soit RWS'(E_1-...-E_n) un sous ensemble de RWS(E_1-...-E_n) défini par restriction aux paires d'objets de RWS'(E_1) X RWS'(E_n); de même pour RWS'(F_1-...-F_p).
Soit $< e_1,e_n > \in$ RWS'(E_1-...-E_n)
Soit $< f_1, f_p > \in$ RWS'(F_1-...-F_p)

Equivalence de chemins : E_1-E_2-...-$E_n \equiv F_1$-F_2-...-F_p est vraie ssi à chaque instant

1. RWS'(E_1-...-E_n) = RWS'(F_1-...-F_p)
2. Il y a une fonction totale bijective g:RWS'(E_1-...-E_n) --> RWS'(F_1-...-F_p) telle que $<f_1,f_p>=g<e_1,e_n>$ ssi $f_1=f(e_1)$ et $f_p=f(e_p)$ et les liens f_1-f_p et e_1-e_n ont la même sémantique"[SPACCAPIETRA 90]

Les correspondances d'inclusion, d'intersection et de disjonction se définissent de façon semblable à celle des éléments TE , TA, attributs.

Synthèse de la méthode(Figure 4.40.)

La **première étape** a pour objectif la définition des assertions de correspondances. Elle est réalisée manuellement ou semi-manuellement par le responsable de l'intégration. Cette étape est qualifiée de semi-manuelle lorsque son responsable est aidé par un outil qui lui permet d'identifier plus facilement les correspondances entre les schémas. Il s'agit généralement d'un outil qui se contente de comparer les noms des éléments des schémas, les domaines des attributs, en bref un outil très primitif et essentiellement basé sur la syntaxe des schémas.

La **seconde étape** consiste à appliquer la règle d'intégration associée à chaque correspondance établie à la phase précédente. Il suffira en effet d'identifier, sur base des éléments et de la relation les mettant en correspondance, la règle d'intégration à mettre en oeuvre. Si un quelconque problème survient dans cette étape, il sera du ressort du responsable du schéma intégré de résoudre le problème.

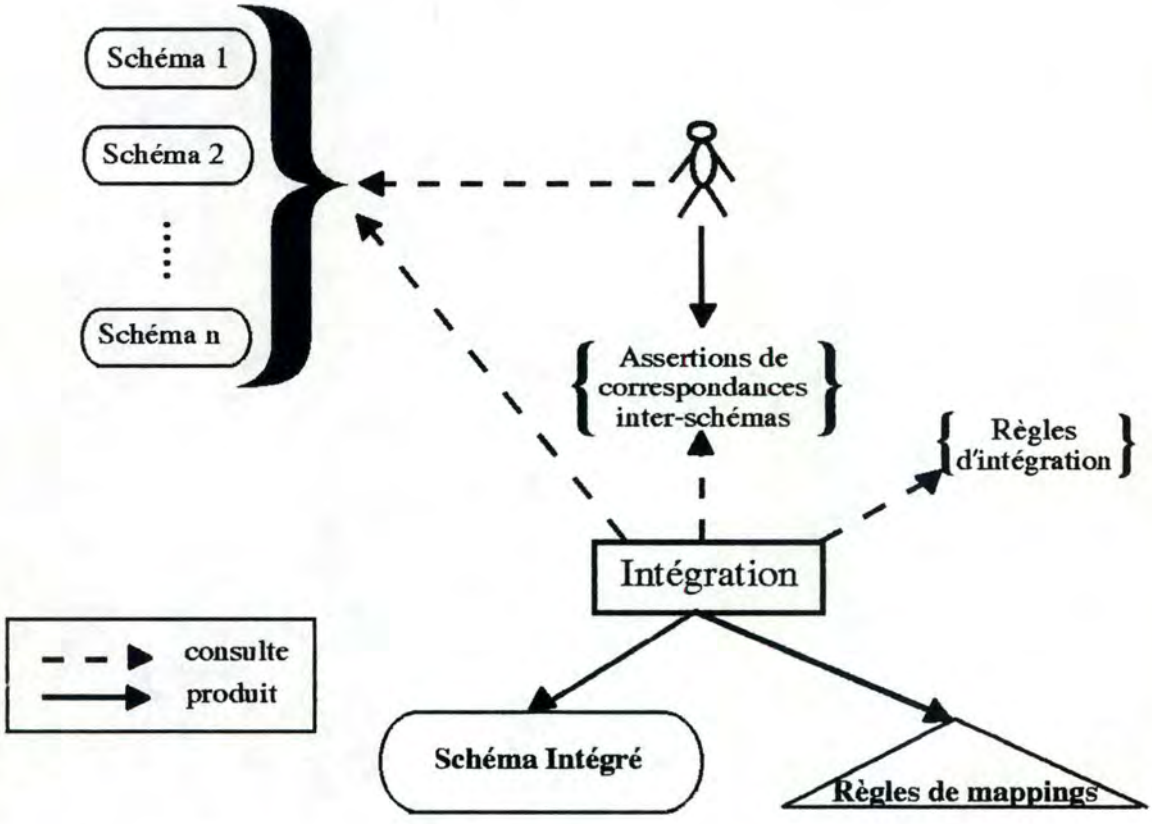


Figure 4.40. : Approche déclarative avec correspondances hétérogènes

Reprenons l'exemple de la figure 4.32, surlequel on peut définir les correspondances suivantes:
Schéma 1. Voiture. Propriétaire ≡ Schéma 2. Personne WCA num-prop = num-p, nom=nom

Schéma 1.Voiture ≡ Schéma 2.Voituret WCA marque=marque, num-chassis=num-chassis
Schéma 1. voiture-voiture.propriétaire ≡ Schéma 2. voiture-possède-personne

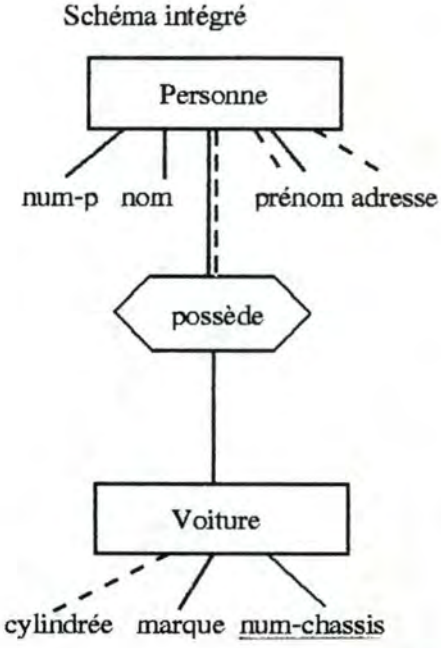


Figure 4.41. : Schéma intégré

La figure 4.41 illustre le schéma intégré basé sur les correspondances précédentes.

On constate donc que cette approche, quelque soit la catégorie de correspondances utilisée, émet l'hypothèse que la personne responsable de la définition des correspondances connaît parfaitement la sémantique originale de chaque schéma. C'est en effet ces correspondances qui définiront le schéma intégré; il est donc évident que la qualité du schéma dépend de ces correspondances. On trouve dans certaines méthodes une étape de vérification de la validité des correspondances pour éviter toute contradiction et pour ajouter les assertions de correspondances qui peuvent se déduire de celles déjà définies par le responsable. On souhaite en effet disposer d'un ensemble **complet** et **cohérent** de correspondances pour que le schéma final présenté soit le plus complet possible, le moins redondant et le plus compréhensible.

4.4.5. Critiques des méthodes d'intégration de schémas

Cette critique sera dirigée essentiellement sur trois axes : le degré d'implication du responsable de l'intégration lors du déroulement de cette phase, la qualité du schéma intégré final et la puissance de la méthode d'intégration.
Pour simplifier la critique, nous dirons que les méthodes procédurales exposées étaient manuelles et que les méthodes déclaratives exposées étaient semi-automatiques.

Concernant le degré d'**implication du responsable** de l'intégration, nous constatons que l'approche procédurale est entièrement l'oeuvre du responsable, du début jusqu'à la fin de cette phase. Il identifie les conflits, les résout et intègre les schémas au moyens des opérateurs de

restructuration ou du langage d'intégration. Il est indispensable pour cette personne de percevoir la solution intégrée avant d'appliquer les opérateurs aux schémas en cours d'intégration. On reprend parfois cette approche comme "l'art de l'intégration". Seule cette personne est capable de décider de la satisfaction de la qualité du schéma intégré final. Il arrête l'intégration quand il le souhaite. L'approche déclarative à base de correspondances homogènes nécessite de la part du responsable d'intégration une sérieuse étape de réflexion notamment pour le conforming et ensuite pour la définition des correspondances. Ensuite sa tâche est beaucoup plus légère puisqu'elle consiste en une intervention occasionnelle en cas de problème lors de l'intégration proprement dite. L'approche déclarative à base de correspondances hétérogènes nécessite une bonne part de réflexion et d'attention puisqu'il s'agit d'établir des correspondances entre des structures hétérogènes. Le responsable devra donc être capable d'identifier parfaitement les conflits et de les exprimer de façon complète et non ambiguë.

Du point de vue de la **qualité du schéma final**, il est difficile de préciser exactement quelle méthode fournit un meilleur schéma. Il est en effet extrêmement difficile de prévoir le schéma intégré final si l'on utilise l'approche procédurale. Il se peut donc que l'on ne choisisse pas au bon moment les bons éléments à intégrer. Le schéma intégré sera différent suivant l'ordre de prise en compte des conflits entre éléments. Dans les approches déclaratives, il est possible que l'ensemble des correspondances établies ne soit pas complet et que le schéma intégré ne soit donc pas le meilleur.

Concernant la **puissance de la méthode d'intégration**, il va de soi que la méthode d'intégration déclarative à base de correspondances hétérogènes est la plus puissante mais rappelons l'hypothèse initiale qui la concerne : il faut établir des règles de mapping entre occurrences de structures distinctes, ce qui pose encore de nombreux problèmes. L'approche procédurale est moins puissante car elle nécessite l'intervention permanente du responsable de l'intégration mais fournit toutefois les règles de mapping automatiquement. L'approche procédurale à base de correspondances homogènes constitue une approche d'une puissance acceptable parce qu'elle nécessite une certaine part d'intervention du responsable de l'intégration et qu'elle fournit des règles de mapping automatiquement.

D'un point de vue général, les méthodes basées sur l'approche déclarative sont encore assez limitées au niveau des correspondances que l'on peut définir. On souhaiterait disposer de correspondances non plus binaires mais n-aires, c'est-à-dire entre un élément d'un schéma et plusieurs éléments d'un autre schémas. Cela nous permettrait de créer des schémas intégrés utilisant les mécanismes de structurations puissants tels que l'agrégation de couverture ou le rôle multi-domaines,...De même, il serait bon que les règles d'intégration identifient certains groupes de correspondances qui peuvent conduire à un structure typique de modélisation plutôt que de traiter correspondance par correspondance. Il faudrait pour cela définir des règles d'intégration non plus associée à une correspondance mais à des combinaisons de correspondances.

Pour conclure cette critique, nous proposons la méthode d'intégration de la section suivante.

4.4.6. Exposé de la méthode d'intégration choisie

En se basant sur les critiques formulées à l'égard des diverses approches, nous proposons une approche mixte c'est-à-dire faisant appel à la fois à l'approche procédurale et à l'approche déclarative en privilégiant toutefois l'approche déclarative. En effet le schéma intégré sera construit sur base d'assertions de correspondances mais il sera éventuellement modifié à l'aide d'opérateur d'intégration pour rencontrer au mieux les caractéristiques attendues d'un tel schéma. Ainsi, trois étapes peuvent être identifiées:

1. Définition des **correspondances** inter-schémas au moyen du modèle de correspondances hétérogènes.

2. **Intégration** proprement dite avec génération automatique des règles de mapping

3. Etape de **restructuration** finale afin d'affiner le schéma intégré et d'obtenir un schéma intégré plus précis : en effet, si l'ensemble des correspondances était incomplet, on envisage de pouvoir affiner le schéma intégré issu de l'étape 3 en utilisation un langage algébrique de restructuration de schéma. Cette étape permettra d'alléger le schéma en utilisant des structures de modélisation plus puissantes telles que l'agrégation de couverture,..

Cette section se contentera d'exposer les principes sous-jacents aux règles d'intégration et le formalisme de description des règles de mapping étant donné que le modèle de description des correspondances a fait l'objet du paragraphe B.2 de la section 4.4.4.

Précisons tout d'abord que la stratégie d'intégration sous-jacente à la méthode utilisée est une stratégie **binaire en échelle** c'est-à-dire que l'on intègre deux schémas à la fois et que le schéma résultant d'une étape d'intégration constitue un des deux schémas de l'étape d'intégration suivante. A titre comparatif, citons les quatre stratégies d'intégration possibles(figure4.42). Il en existe typiquement deux grandes catégories, les stratégies binaires et les stratégies n-aires.
Les **stratégies binaires** travaillent par paire de schémas, ce qui signifie que deux schémas sont inétgrés à la fois et ce processus sera répété plusieurs fois de suite. Il existe deux techniques d'application, la technique **en échelle** et celle **équilibrée**.
Les stratégies n-aires travaillent sur l'ensemble des schémas à intégrer. Deux techniques ont été développées, l'une **itérative** et l'autre **en un coup**.
La stratégie binaire a l'inconvénient d'obliger le responsable de l'intégration à définir pour chaque paire de schémas des assertions de correspondances et à recommencer ce processus plusieurs fois. Ce processus de définition de correspondances et d'intégration est donc répétitif mais offre l'avantage de mieux cerner les correspondances car il apparaît naturellement que

définir des correspondances entre n (n>2) schémas est une tâche complexe tout comme celle de définir un modèle de description de correspondances de cette sorte.
La stratégie en échelle offre l'inconvénient de séquentialité du processus d'intégration: il faut attendre que deux schémas soient intégrés avant d'en intégrer deux autres. Par contre la stratégie balancée permet de travailler en parallèle. Toutefois la stratégie en échelle permet d'intégrer les schémas selon un ordre laissé au responsable de l'intégration. L'ordre de prise en compte des schémas à intégrer peut en effet jouer un rôle important dans la résultat final or il apparaît couramment dans une organisation que des BD soient privilégiées par rapport à d'autres.

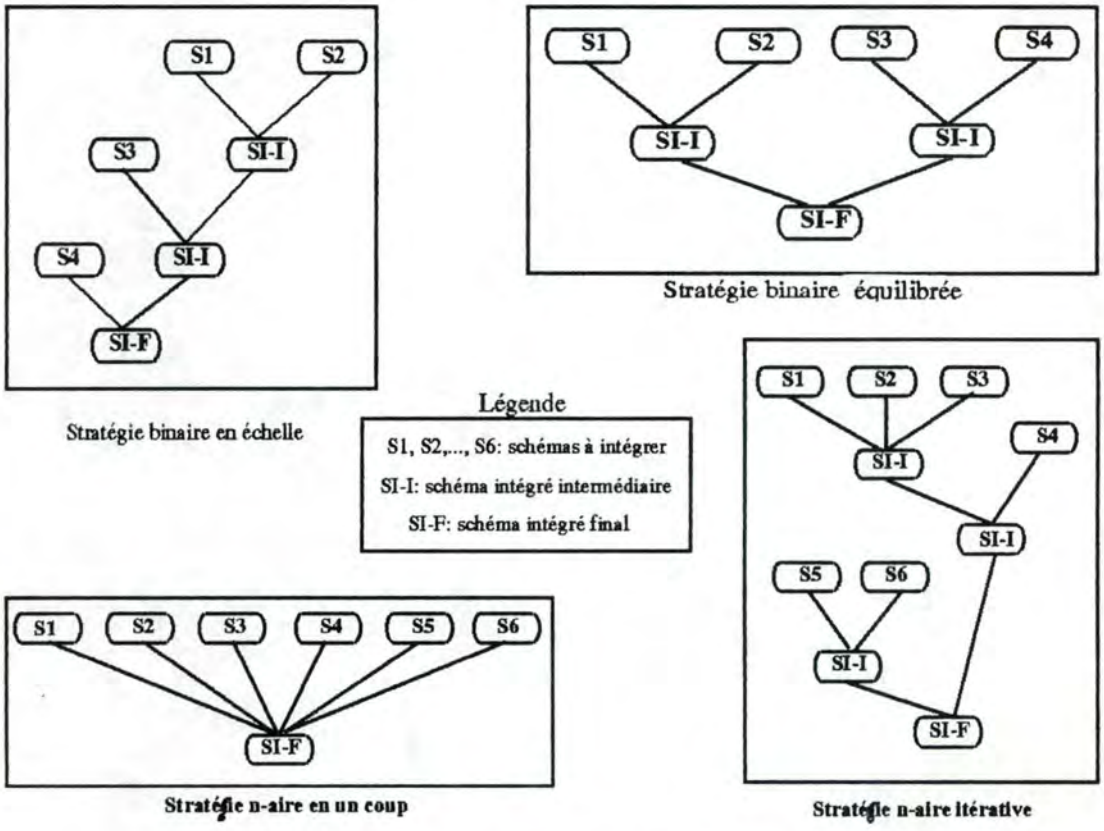


Figure 4.42. : Stratégies d'intégration

Sans entrer dans le détail, nous citerons les principes régissant l'intégration de deux schémas. On trouvera en annexe une liste non exhaustive de règles d'intégration accompagnées des règles de mapping.

Les cinq **principes généraux** sont les suivants:

1. Deux éléments de même structure (TE,TA) en correspondance d'équivalence donne naissance à un seul élément de cette même structure.

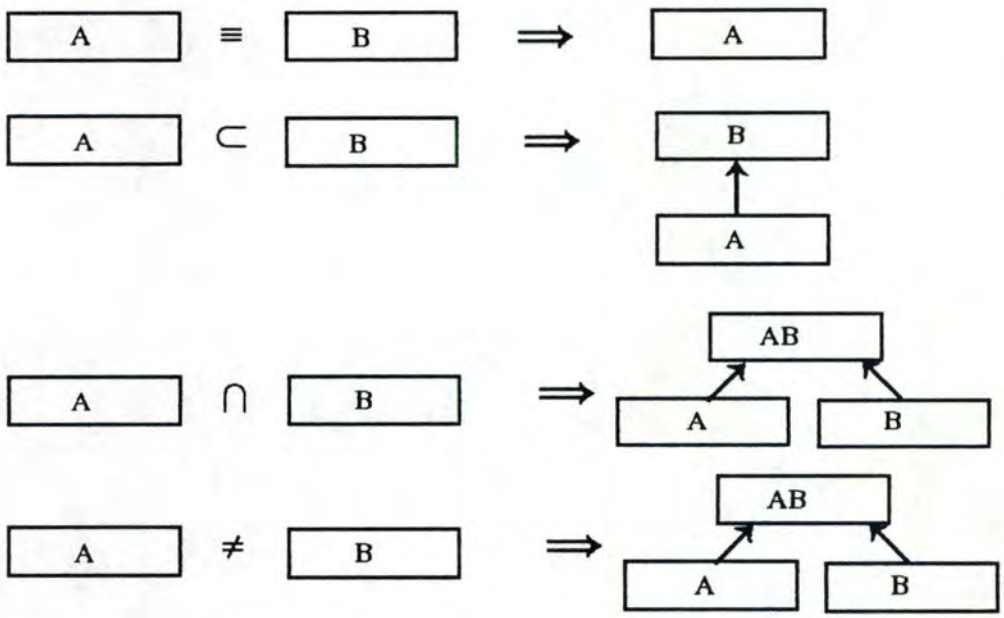
2. Un TE et un TA en correspondance d'équivalence donne naissance à un seul élément structuré en TE. En effet les TE ont une existence autonome ce qui n'est pas le cas pour les TA qui sont obligatoirement dépendants d'un TE d'un ou plusieurs TE.

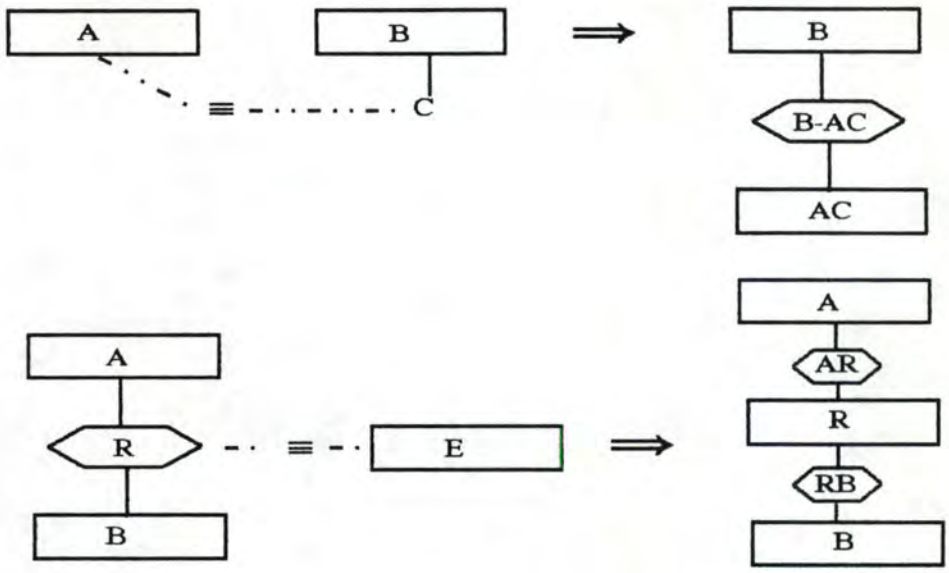
3. Tout attribut n'apparaissant pas dans une clause WCA, sera repris tel quel dans le schéma intégré; quant aux autres attributs ils seront soit repris sous un seul attribut ou individuellement suivant les relations les unissant.

4. Le résultat de l'intégration de deux chemins dépendra de la présence d'information dans les chemins : pour fusionner deux chemins en un seul il faut que l'on puisse réduire un des deux chemins à l'autre sans perte d'informations.

5. Face à deux structure en conflits de description au niveau des cardinalités d'attributs ou de rôles on optera toujours pour la représentation la moins restrictive

De façon abstraite et non exhaustive, on peut définir les règles d'intégration suivantes:





On trouvera un exposé formel des règles d'intégration en annexe 2.

Les **règles de mapping** associées sont doubles pour chaque correspondance :

- **Mapping syntaxique:** celui-ci établit la correspondance entre les concepts du schéma intégré et ceux des schémas initiaux pour le traitement des requêtes globales et plus spécialement leur décomposition (Cfr section 4.5)

- **Mapping des occurences :** - spécifie la manière de construire les **occurrences du schéma intégré au départ des occurrences des schémas initiaux** (S1,S2) SI.élément = Integrate-join (S1.élément, S2.élément, correspondances d'attributs)

  - spécifie la manière de construire les **occurrences des schémas locaux au départ des occurrences du schéma global**

    S1.élément = f(SI.élément)

    S2.élément = g(SI.élément) avec f,g

    (combinaison d')opérateur(s) algébrique(s) EA/E

Comme on l'a signalé le modèle de définition des règles de mapping est d'une telle complexité que peu de modèle complet existe à l'heure actuelle. C'est pourquoi seule quelques règles de mapping sont disponibles en regard des règles d'intégration proposées en annexe.

Les opérateurs de restructuration finale reprennent en plus des opérateurs du langage de définition de schémas L-EA/E les opérations de "rename" d'un élément quelconque, d'effacement ("delete") d'un élément ainsi qu'un certain nombre de transformations tels que

transformer un TA en TE, une structure avec des TE et des TA en agrégation de couverture(Figure 4.44) ou en rôle multi-domaines,...

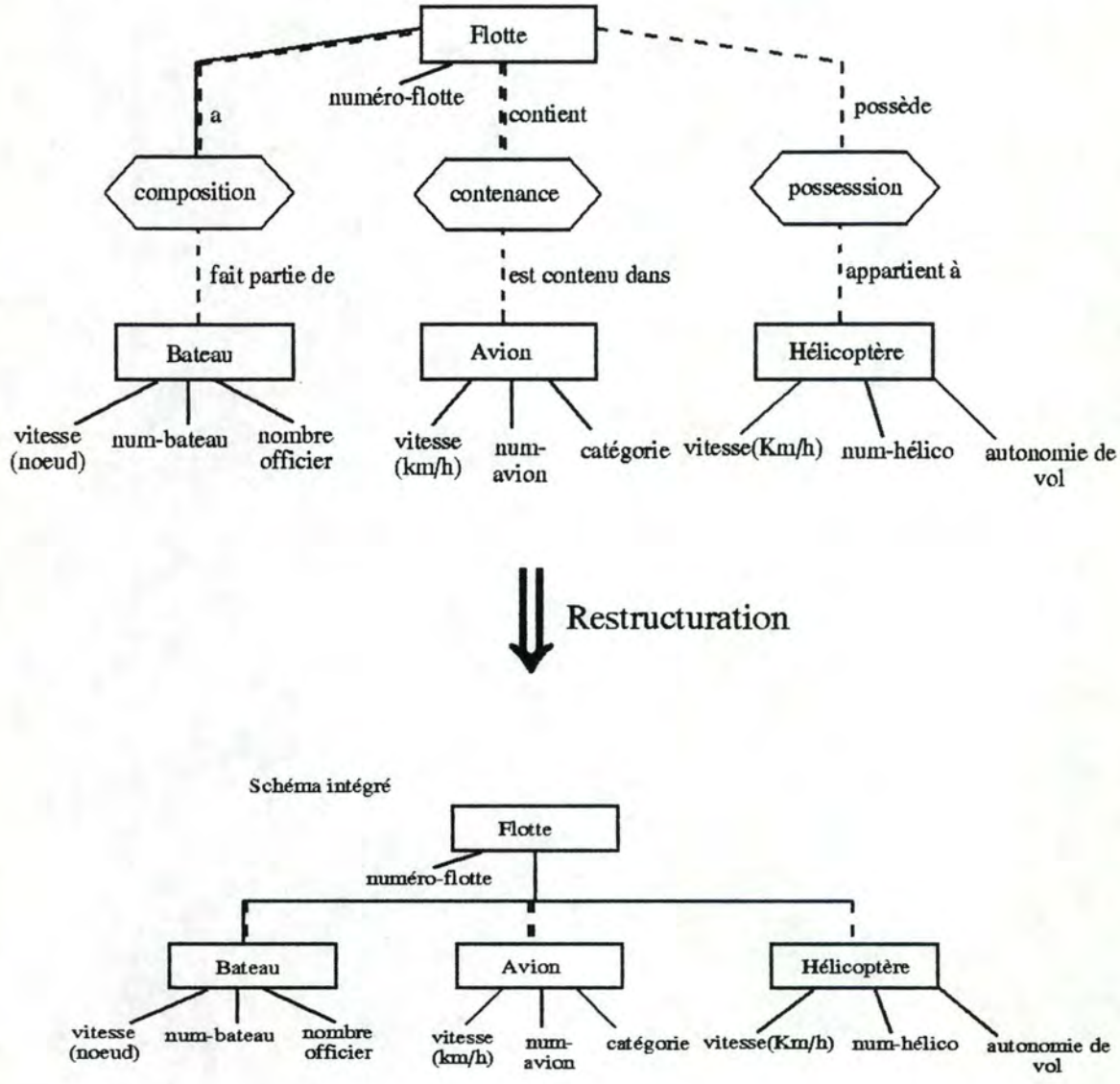


Figure 4.44.: Exemple de restructuration

4.4.7. Problème d'intégration de données

Ce problème est en majeure partie résolu par les règles de mapping définies lors de l'intégration des schémas. Ces dernières se chargent notamment d'éliminer la redondance des données.

En supposant que l'on est à intégrer les résulats fournies par deux requêtes exécutées sur des BD dont les schémas sont ceux de la figure 4.32. La requête globale exigeait d'obtenir l'ensemble des voitures(num-chassis,cylindrée, marque) des propriétaires de nom "DUPONT" et

d'adresse"NAMUR" . Les résultats des sous-requêtes (cfr. 4.5) sont les suivants:

Schéma 1

Ensemble des occurences de **Voiture** dont la cylindrée dépasse 1300 cm^3 et dont le propriétaire a pour nom"DUPONT".

Num-chassis	Cylindrée	Marque
XDE2152	1500	VW
FRD5896	2000	Peugeot
JUU9865	1400	Renault
AZE2658	1300	Volvo

Schéma 2

Ensemble des occurences de **Voiture** dont la PERSONNE propriétaire a pour nom"DUPONT" et adresse "NAMUR"

Num-chassis	Marque
XDE2152	VW
FRD5896	Peugeot
AZE2658	Volvo
TRE9999	Citroën

Lorsqu'on examine ces deux résultats, on constate que ni l'un ni l'autre ne répond à la requête globale. Par contre les éléments communs à ces deux résulats correspondent bien à la requête globale. Il nous suffira donc de retirer de chacun de ces résulats les attributs identifiants en communs et de les compléter par les attributs des deux résulats. Cela revient à faire une *jointure* [1] des résulats sur base de l'identifiant Num-chassis suivi d'une *projection*[1] sur les attributs exigés ce qui nous forunit le résultat intégré suivant:

Ensemble des occurences de **Voiture** dont la cylindrée dépasse 1300 cm^3 et dont le propriétaire a pour nom"DUPONT" et adresse "NAMUR"

Num-chassis	Cylindrée	Marque
XDE2152	1500	VW
FRD5896	2000	Peugeot
AZE2658	1300	Volvo

Nous avons pu constater l'importance qui doit être accordée aux correspondances entre les identifiants lors de la définition des assertions de correspondances. Cela est indispensable pour réaliser l'intégration des données.

Toutefois subsiste le problème des données conflictuelles autres que celles issues des différences d'échelles. Ces dernières sont en effet résolues par **algorithmes** ou **tables de correspondances** qui sont liés aux règles de mapping.

1. opérateur algébrique défini en annexe 1

Les données conflictuelles sont des données qui devraient être identiques mais qui ne le sont pas. A titre d'exemple, supposons que la voiture de Num-chassis "XDE2152" soit de marque "VW" dans le schéma 1 et "Mercedes" dans le schéma 2. Nous avons exprimé les origines possibles de ce problème et cela nécessite un traitement particulier. Dans une telle situation, plusieurs solutions sont possibles:

1. Ne **rien** fournir à l'utilisateur sous prétexte que l'on ne fournit que des données dont il est certain qu'elles soient correctes.

2. On fournit à l'utilisateur l'ensemble des données conflictuelles en l'avertissant du problème par un "**warning**" et ce sous prétexte que l'utilisateur est capable de discerner le bon du mauvais.

3. Si l'intégration porte sur plus de deux données conflictuelles, on travaille sur base de la **majorité**. La donnée la **plus** fréquemment trouvée est donc fournie. Cela se base sur le principe que la vérité est issue de la majorité.

4. Si l'intégration porte sur plus de deux données conflictuelles, on travaille sur base de la **minorité**. La donnée la **moins** fréquemment trouvée est donc fournie. Cela se base sur le principe que la vérité est issue de la minorité. Il se peut qu'une modification locale ait été produite sur une seule BD.

5. Si le type des données le permet, une **moyenne** des valeurs peut être établie.

6. Installer un **dateur** sur chaque donnée qui fournit le jour et l'heure de la dernière modification de chaque donnée. Il suffirait de consulter la donnée la plus récemment modifiée. C'est la solution idéale mais irréaliste.

Il faudrait donc spécifier une **règle de préférence** à appliquer en cas de conflits de données. Cette règle de préférence sera différente en fonction du contexte dans lequel on se situe. Il existe en effet des milieux où aucune erreur n'est admissible; dans ce cas on préfèrera ne pas donner de réponse plutôt que de fournir une réponse incomplète ou probablement erronée. Ce choix se situant au delà de nos préoccupations en matière de BDH, nous laisserons la question ouverte.

A présent que le problème de l'intégration a été analysé, passons au problème du traitement des requêtes.

4.5. Traitement des requêtes globales

4.5.1. Introduction

Pour rappel, une requête globale est une requête qui est définie sur le schéma fédéré de notre architecture et qui porte indirectement sur un sous-ensemble ou la totalité des bases de données locales. Cette requête ne spécifie pas l'endroit où les données réelles sont stockées. Si l'on reprend l'exemple de la figure 4.22, une requête globale serait par exemple "donner l'ensemble des noms des propriétaires de voiture de marque 'VW'? " et se traduirait en langage L-EA/E comme suit:

SELECT nom
FROM Propriétaire, Voiture
WHERE Propriétaire possède Voiture AND marque="VW"

On constate que cette requête ne fait référence qu'aux concepts du schéma fédéré (ou schéma intégré). Cette requête sera donc décomposée en sous-requêtes qui seront exécutées ultérieurement sur les SBD locaux. Cette décomposition devra se faire en respectant les concepts de chacun des schémas exportés auxquels se réfèreront les sous-requêtes. Les règles de mapping établies lors de l'intégration des schémas exportés nous permettront d'établir ces sous-requêtes. Cette phase de décomposition se contentera de déduire de la requête globale un ensemble de sous-requêtes sur les schémas exportés, ce qui signifie qu'elles ne sont pas encore exécutables par les SGBD locaux et qu'elles doivent dès lors faire l'objet de transformations (section 4.3.2) afin d'être exécutées réellement. Cela est la conséquence du caractère virtuel des différents niveaux de schémas de notre architecture, à l'exception du niveau local qui a un caractère réel vu que les données y sont stockées.

Ainsi pour l'exemple cité on aura les sous-requêtes suivantes :

Schéma 1 : SELECT nom
FROM Propriétaire
WHERE voiture. marque = "VW"

Schéma 2 : SELECT propriétaire.nom
FROM Voiture
WHERE marque = "VW"

A cette phase de décomposition est associée une phase d'optimisation. En effet, l'exécution simple des sous-requêtes au niveau local et l'intégration au niveau global des résultats pourraient conduire à des temps de réponse considérables, ce qui est peu souhaitable lorsque l'on travaille en mode interactif. C'est pourquoi on essaiera étant donné le contexte dans lequel on travaille de tirer parti de certains facteurs.

4.5.2. Traitement d'une requête

La phase complète de traitement d'une requête globale consiste en deux parties: la **définition d'un plan d'exécution** et l'**exécution** de celui-ci.
Il s'agit tout d'abord de définir un **plan** spécifiant l'ensemble des tâches à effectuer ou à faire effectuer par les SBD locaux. Tout plan d'exécution devra vérifier les deux exigences suivantes:

- il devra produire le résultat escompté; il doit donc être **fidèle** à la requête globale

- il doit **tendre vers une solution optimale.** Il devra donc minimiser une fonction de coût qui incorporera une variable de consommation de ressources.

Etant donné la difficulté de vérifier ces deux exigences simultanément, elles seront typiquement isolées en deux étapes séquentielles, la décomposition et l'optimisation globale. Cette dernière sera fonction d'un certain nombre de facteurs que nous exposerons ci-après.
Un fois le plan d'exécution établi, il sera mis à exécution sous le contrôle du SGBDH.

Le traitement des requêtes globales pose un problème particulier, que ce soit des requêtes de consultation ou de mise-à-jour. En effet, supposons que l'exécution d'une requête globale fasse appel à plusieurs (5 par exemple) SBD locaux. Cinq résultats (données fournies ou mise-à-jour effectuées) sont donc attendus. Si pour une raison quelconque (panne d'un SBD local,...) une des sous-requêtes ne pouvait être exécutée alors le résultat exigé par l'utilisateur pour la requête globale serait incorrect: soit les données qui lui sont fournies sont incomplètes, soit les mise-à-jour n'ont pas toutes été réalisées comme demandées. On peut proposer plusieurs solutions face à ce type de problème :

Pour les opérations de consultation, l'instauration d'une étape de synchronisation des résultats fournis par les SBD locaux permettrait de voir s'il y a eu problème dans un des SBD locaux. On peut dès lors décider de ne fournir aucun résultat à l'utilisateur sous prétexte qu'il serait incomplet et donc incorrect. Soit on décide de lui fournir les résultats disponibles en prenant soin de l'avertir de la situation.

Pour les opérations de mise-à-jour, on peut envisager les alternatives suivantes:

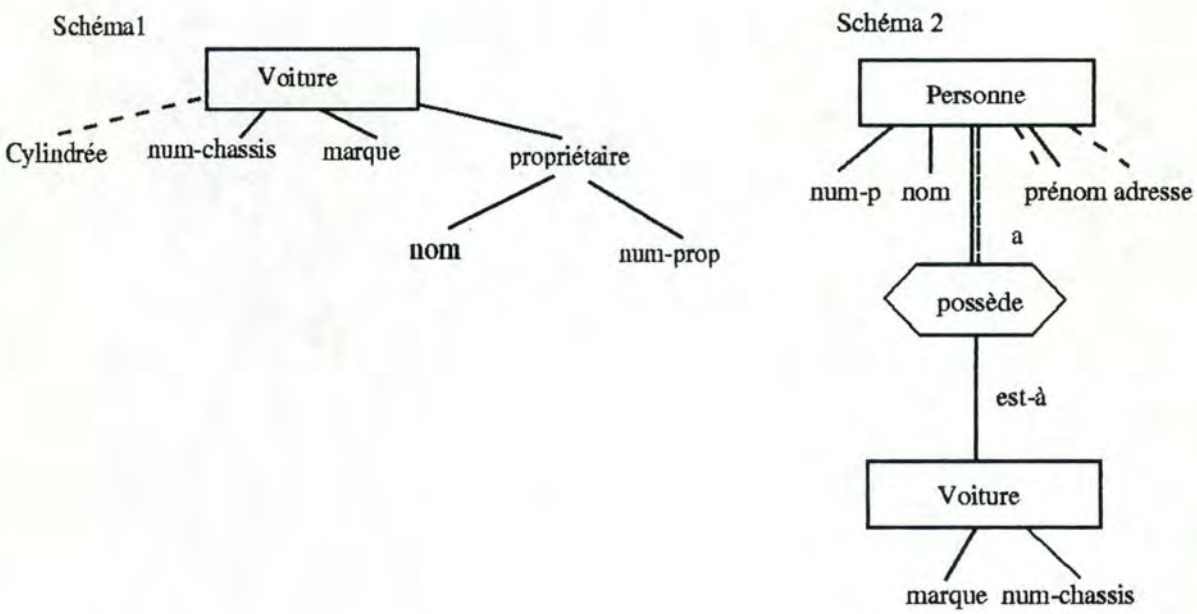
- si un quelconque problème intervient au niveau local, il faudrait être en mesure d'annuler les modifications qui ont été appliquées sur tous les autres SBD locaux
- une autre solution consisterait à conserver les mise-à-jour réellement effectuer et à faire en sorte que le SGBDH puisse conserver ce fait afin d'en tenir compte dans le traitement des requêtes à venir. En effet, si le SGBDH sait qu'une ou plusieurs BD n'ont plus être modifiées correctement, il pourra établir des règles de gestion des conflits de

données. Toutefois, le SGBDH devra garder en mémoire les modifications qui n'on pu être réalisées pour les exécuter dès que possible. On pourrait en effet accepter cette solution étant donné l'autonomie des SBD qui est elle-même source d'hétérogénéité des données; la cause en est la présence de mises-à-jour locales non repercutées au niveau global.

A présent analysons de plus près le problème de décomposition des requêtes globales.

4.5.3. Décomposition

Si pour réaliser cette phase on se contente des règles de mapping établies lors de l'intégration, on est sûr que la décomposition sera fidèle.
Il suffira donc pour chaque requête globale de conserver son format afin de définir les sous-requêtes et d'adapter les arguments en consultant les règles de mapping. Ces dernières permettront d'identifier les schémas concernés par la requête globale et ainsi de sélectionner les SBD locaux dont les services seront exigés.
Concernant ce problème de décomposition, nous pouvons renvoyer le lecteur au contexte des bases de données réparties. En effet, dans ce contexte on dispose d'un schéma global et de plusieurs BD **homogènes** mais on y travaille avec **un** seul modèle de données et **un** seul langage de définition/manipulation. C'est pourquoi l'on retrouve ce problème particulier.
Prenons l'exemple de la figure 4.45 qui présente deux schémas composants et le schéma intégré correspondant.



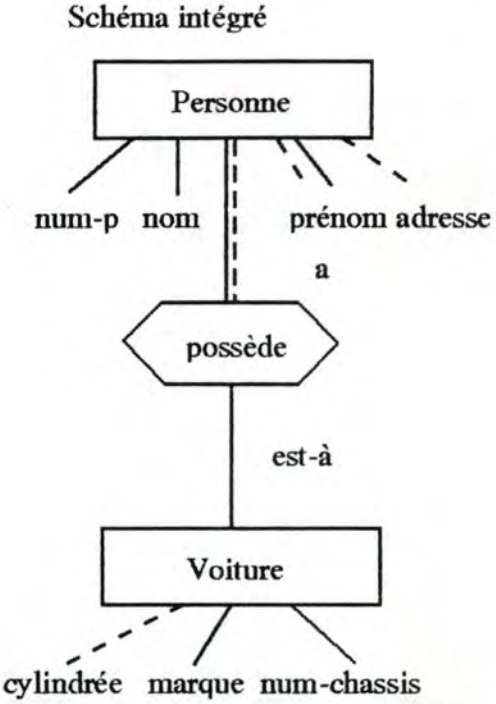


Figure 4.46: Deux schémas composants et le schéma intégré correspondant

Les règles de mapping syntaxique entre ces schémas sont les suivantes:

[1] Personne = S1.Voiture.propriétaire WCA num-p=num-prop, nom=nom
[2] Personne = S2. Personne WCA num-p=num-p, nom=nom, prénom=prénom, adresse=adresse
[3] Voiture = S1.Voiture WCA num-chassis=num-chassis, marque=marque, cylindrée=cylindrée
[4] Voiture = S2.Voiture WCA num-chassis=num-chassis,marque=marque
[5] Possède = S2.possède
[6] Personne - a -Voiture = S1.Voiture.propriétaire - Voiture
[7] Personne - a -Voiture = S2.Personne - a - Voiture
[8] Voiture- est-à - Personne = S1.Voiture -Voiture.propriétaire
[9] Voiture - est-à -Personne = S2.Voture - est-à - Personne

Soit la requête sur le schéma intégré de la figure 4.46. :

```
SELECT marque,num-chassis, cylindrée
FROM Voiture, Personne
WHERE nom= "DUPONT" AND adresse="NAMUR"
            AND  Personne a Voiture  AND cylindrée > 1300
```

On décomposera cette requête comme suit :

Sur le schéma S1:

SELECT marque,num-chassis, cylindrée = > [3] SELECT marque, num-chassis,cylindrée
FROM Voiture, Personne = > [1][3] FROM Voiture
WHERE nom= "DUPONT" = > [1] WHERE propriétaire.nom = "DUPONT"

AND adresse="NAMUR"	= >	[1]
AND Personne a Voiture	= >	[6]
AND cylindrée > 1300	= >	[3] AND cylindrée > 1300

La règle [1] ne fournit pas de correspondant pour l'attribut "adresse";cela explique le vide dans la requête.
La règle [6] fournit un correspondant sans intérêt étant donné qu'il s'agit d'un chemin entre un attribut et son TE père. Cela explique l'absnce de correspondant dans la requête.

Sur le schéma S2 :

SELECT marque,num-chassis, cylindrée	= >	[4] SELECT marque, num-chassis
FROM Voiture, Personne	= >	[2][4] FROM Voiture, Personne
WHERE nom= "DUPONT"	= >	[2] WHERE nom = "DUPONT"
AND adresse="NAMUR"	= >	[2] AND adresse="NAMUR"
AND Personne a Voiture	= >	[7] AND Personne a Voiture
AND cylindrée > 1300	= >	

La règle [4] ne fournit pas de correspondant pour l'attribut "cylindrée"; cela explique le vide dans la requête.
Chacune de ces requêtes exécutées indépendamment l'un de l'autre va fournir un résultat qui n'est pas celui de la requête globale. C'est là qu'intervient l'intégration des données dont il a été question à la section 4.4.7.
On a déjà insisté sur l'importance des correspondances entre identifiants dans le sous-chapitre concernant l'intégration. Nous avons montré son incontournable nécessité pour intégrer les données. C'est pourquoi, lorsque l'on décomposera une requête, on devra toujours retrouver dans les arguments du résultat de chacune des sous-requêtes au moins un attribut identifiant. Dans le cadre de notre exemple ci-dessus, si l'attribut "Num-chassis" n'avait pas fait partie des arguments du résultat de la requête globale, on aurait quand même trouver cet attribut dans les sous-requêtes.

Le traitement de certaines requêtes portant sur des structures de modélisation plus complexes telles que la généralisation-spécialisation, l'agrégation de couverture et le rôle muti-domaines pourra apparaître plus particulier. Mais si les règles de mapping sont correctement établies il n'y aura aucun problème. Le problème réside, comme nous l'avons signalé, dans la définition de ces règles de mapping.

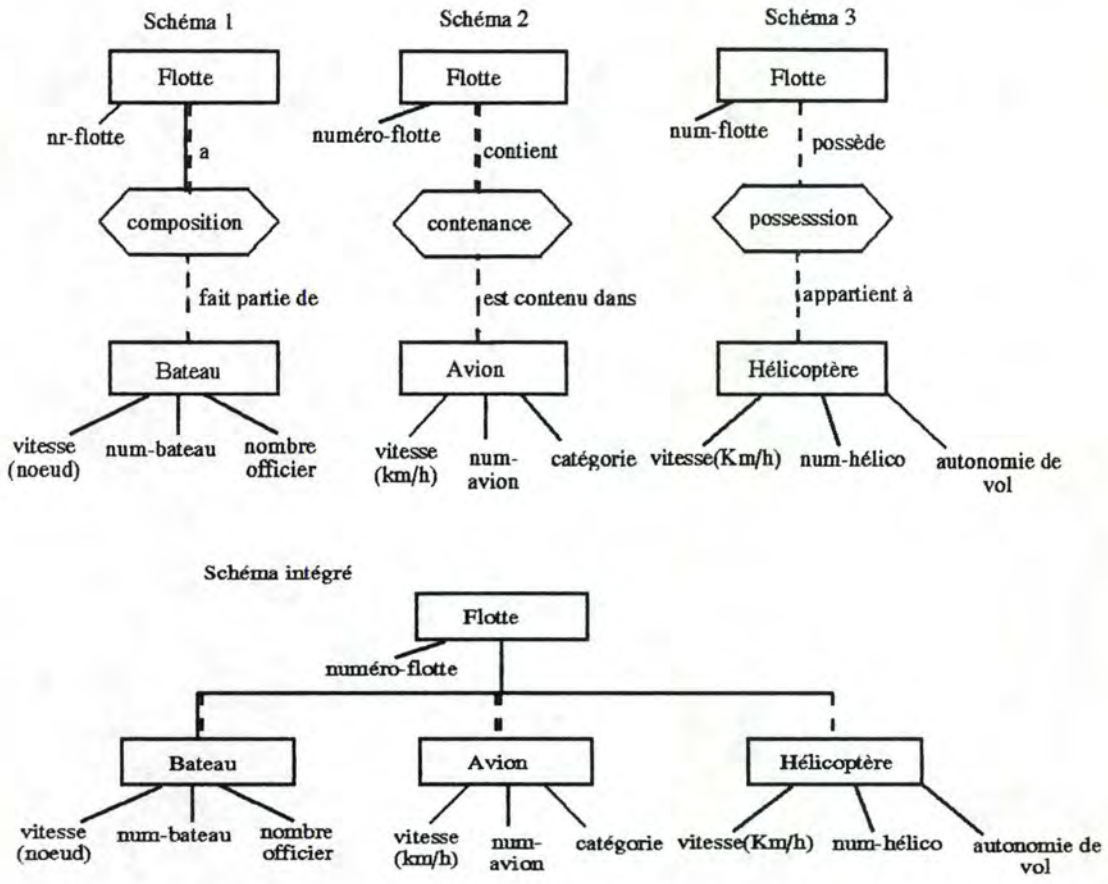


Figure 4.46.: Exemple de schémas avant et après intégration

Soit la requête globale posée sur le schéma intégré: "Donner le contenu de la flotte de numéro 'X325'" et définie comme suit:

SELECT num-bateau, num-avion, num-hélico **FROM** Bateau,Avion,Hélicoptère
WHERE Bateau IN Flotte AND Avion IN Flotte
AND Hélicoptère IN Flotte
AND numéro-flotte="X325"

Cette requête devra être décomposée comme suit:

SELECT num-bateau FROM Bateau WHERE Bateau fait partie de Flotte AND nr-flotte =" X325"
SELECT num-avion From Avion WHERE Avion est contenu dans Flotte AND numéro-flotte =" X325"
SELECT num-hélico From HélicoptèreWHERE Hélicoptère appartient à Flotte AND num-flotte =" X325"

4.5.4. Optimisation en BDH vs. optimisation en BD centralisée

Le problème d'optimisation est plus complexe en BDH qu'en BD centralisée pour plusieurs raisons:

1. On a une multitude de SBD **hétérogènes** dont les différences se rencontrent au niveau des services des SGBD, des primitives d'accès, des langages d'interrogation et des formats de données.

2. Ces SBD locaux sont **autonomes** ce qui implique :

- l'exécution de requêtes locales indépendamment des requêtes globales

- l'impossibilité de divulguer des informations sur le coût local d'exécution de certaines opérations

- la liberté du système local de choisir les chemins d'accès pour exécuter une requête issue du SBDH.

3. Ces SBD sont pour la plupart **localisés à des endroits différents** ce qui implique l'existence de moyens de communication.

4. **Redondance** des données au travers des divers SBD locaux. De plus cette redondance peut être de type conflictuelle: par exemple deux attributs identiques de deux schémas différents présentent des valeurs différentes alors qu'elles devraient être identiques.

On a donc beaucoup plus de facteurs à prendre en compte lors de la définition d'un plan optimal d'exécution.

L'optimisation du plan d'exécution est nécessaire pour obtenir un temps de réponse acceptable. Celui-ci sera constitué des facteurs suivants:

Supposons que la requête globale requiert les services de i SBD locaux sur n et qu'elle se limite à une requête par SBD. Le schéma de la figure 4.47 reprend les différents temps qui constituent le temps de réponse à cette requête.
Ce schéma fait l'**hypothèse** que la traduction des sous-requêtes et des données est réalisée à l'endroit où sont situés les SBD locaux. Cela explique les temps de transfert entre le processeur responsable de la phase de décomposition-optimisation et ceux chargés de la traduction des sous-requêtes et des données. On aurait tout aussi bien pu envisager de regrouper les processeurs chargés de la traduction au même niveau que celui gérant la décomposition-optimisation. Mais le transfert aurait toutefois dû se faire entre ces processeurs. La seule différence porte sur le contenu du transfert puisque dans ce dernier cas il s'agirait de transférer des sous-requêtes directement exécutables par les SGBD locaux.

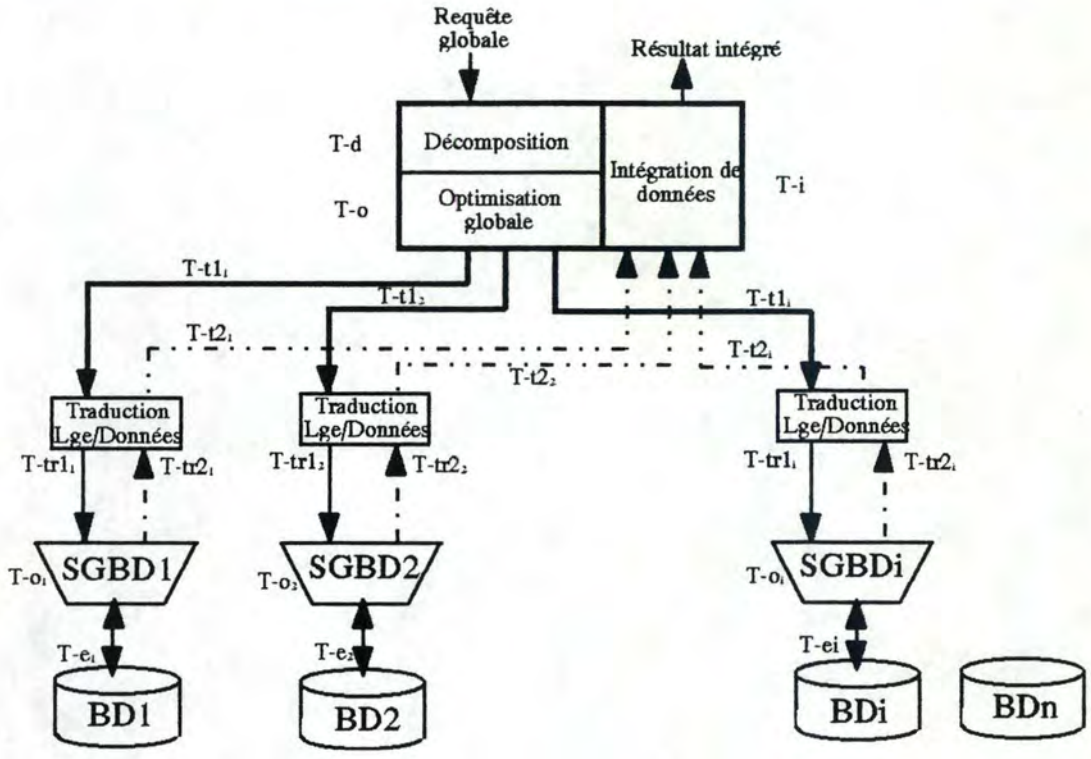


Figure 4.47. : Décomposition du temps de réponse pour une requête globale

- Temps de décomposition de la requête globale (**T-d**)

- Temps d'optimisation de la requête globale (**T-o**)

- Temps de transfert de chacune des i requêtes entre le SGBDH et le Système Local supportant le SBD local correspondant (**T-t1** $_i$)

- Temps de traduction de chacune des requêtes dans le langage local correspondant (**T-tr1** $_i$)

- Temps d'optimisation locale de chacune des requêtes locales (**T-o**$_i$). Certains SGBD ne proposent pas cette phase, toutefois on pourrait envisager de réaliser cette étape en dehors du SGBD.

- Temps d'éxécution de chacune des requêtes locales y compris les temps d'accès à la BD(**T-e**$_i$)

- Temps de traduction de chacun des résultats dans le langage canonique(**T-tr2** $_i$)

- Temps de transfert de chacun des résultats vers le SGBDH (**T-t2**$_i$)

- Temps d'intégration des données (**T-i**)

On a ainsi différents facteurs à prendre en compte pour définir une fonction de coût de traitement d'une requête globale. Toutefois il est possible de regrouper l'ensemble des facteurs influents en trois catégories : les **facteurs de communication** et les **facteurs de calcul**(CPU) et les **facteurs d'accès BD.**

Le modèle de coût devra être tel qu'il prend en compte le plus de facteurs possibles. Néanmoins le temps d'exécution de cette algorithme ne devra pas dépasser le temps de réponse de la stratégie de résolution la plus mauvaise. On préférera en effet obtenir une sorte d'optimum dans des délais de quelques secondes plutôt que d'attendre plusieurs heures ou jours afin d'obtenir l'optimum par excellence.

Maintenant que nous avons montré l'intérêt de cette phase d'optimisation, nous allons analyser les approches de solutions que l'on peut adopter pour définir un algorithme d'optimisation. Nous travaillerons par analogie avec les bases de données distribuées homogènes et l'on adaptera les solutions dans la mesure du possible.

Tout algorithme d'optimisation est caractérisé par deux éléments:

1. **Le critère de base de l'optimisation**: il s'agit des ressources dont la consommation sera minimisée par l'algorithme d'optimisation et qui est en relation avec le temps de traitement d'une requête. La section 4.5.5 présentera les alternatives dans ce domaine.

2. La **stratégie de sélection du plan d'exécution**: vu que plusieurs plans d'exécution peuvent être définis sur base d'une requête globale, l'algorithme doit disposer d'une stratégie ou technique de sélection pour déduire le meilleur plan. La section 4.5.6. fournira les diverses stratégies possibles.

La section 4.5.7 montrera également l'influence de l'architecture d'implantation physique du SBDH sur cette phase d'optimisation globale.

4.5.5. Critères d'optimisation globale

Deux critères d'optimisation peuvent être pris en compte, le premier porte sur le calcul ou le temps CPU tandis que le second porte sur le transfert entre SBDH et SBD locaux.
Le facteur "temps d'accès" ne constitue pas un critère d'optimisation globale du fait du respect due à l'autonomie des SBD locaux.

A. Critère "Temps CPU"

Le temps de traitement CPU des requêtes globales apparaît en effet considérable. Il suffit de regarder le schéma de la figure 4.46. pour s'en convaincre. La décomposition, l'optimisation, la

traduction des requêtes et des données sont des processus qui peuvent exiger de grandes quantités de ressources.
Il est donc possible de développer un algorithme basé sur le critère de minimisation de la consommation de cette ressource. L'hypothèse de départ d'un tel algorithme est la suivante:" Le coût de traitement CPU est plus élevé que celui des communications". Ainsi cet algorithme aura pour objectif de **maximiser le parallélisme** lors du traitement des sous-requêtes. Il veillera donc à minimiser le temps CPU sans tenir compte du coût des communications. Cet algorithme ne cherchera donc pas à éliminer les données redondantes avant de les transférer.
Dans le cadre des BD distribuées homogènes, on peut tirer profit de la duplication des données. On soumettra une même requête à plusieurs SBD locaux stockant les mêmes données en prenant soin d'exiger d'eux la réponse à une partie de la requête initiale. Par exemple, on dispose de trois BD avec les Clients (numéro, nom, prénom, adresse). On peut décider de rechercher le nom dans la première BD, le prénom dans la seconde et l'adresse dans la dernière sur base d'une requête de recherche de ces éléments et fournissant le numéro de client. Cela permet d'améliorer les performances mais n'est toutefois pas applicable au contexte des BDH du fait de l'autonomie des SBD locaux; ceci peut en effet engendrer l'existence de données conflictuelles et on ne peut dès lors pas user de la redondance des données sans faire d'hypothèse restrictive sur l'état des BD locales.

B. Critère " Temps de transfert"

Etant donné la distribution des BD locales et leur hétérogénéité, on est face à un problème de communication qui peut nécessiter l'utilisation de moyens de télécommunication (satellite, réseau large bande, réseau téléphonique,...). Etant donné le coût des communications on peut envisager d'optimiser les requêtes en vue de minimiser ces coûts et ce en **limitant le volume de transfert de données.**
Les algorithmes basés sur ce critère minimiseront le coût de transfert au détriment du coût du CPU.
La solution généralement proposée est de réaliser des opérations locales (élimination d'attribut non repris dans la requête,...) et d'intégration intermédiaires. Pour cette dernière, il s'agit sur base des résultats de deux sous-requêtes de réaliser l'intégration au niveau composant de notre architecture, c'est-à-dire lorsque les données sont représentées dans le même format. Seul le résultat de cette intégration sera transmis au SGBDH. Ces opérations d'intégration sont généralement appelées *"semi-jointure"*. Une semi-jointure consiste sur base de deux résultats présentant des données communes à fournir un résultat non redondant c'est-à-dire de volume moindre. Le problème essentiel porte sur le choix des résultats à intégrer. Plusieurs semi-jointure sont généralement possibles mais il faut ne retenir que celles qui nous apporteront un avantage.

Ces deux types d'algorithmes sont tous deux valables. Il faut en effet tenir compte de la situation réelle dans laquelle on doit développer le SBDH. Si les SBD locaux sont tous situés dans le même voisinage alors il est évident que les coûts de communications ne sont pas les plus élevés. Par contre s'ils sont situés à de grandes distances l'un de l'autre ces coûts auront une proportion

tout à fait différente. Le temps CPU quant à lui est important à prendre en compte suivant que les systèmes hardware supportant les SBD locaux sont fortement utilisés ou pas du tout. La puissance et la charge de travail des CPU locaux sont les facteurs déterminants dans ce cas.
Le choix de l'algorithme repose donc sur le cas particulier à traiter. La meilleure solution consisterait à prendre en compte les deux critères simultanément afin de limiter au maximum la consommation de ressources sans pour autant oublier le compromis entre complexité de l'algorithme d'optimisation et qualité du plan d'exécution.

4.5.6. Les stratégies de sélection du plan d'exécution

On peut proposer trois stratégies de sélection:

A. Stratégie basée sur une énumération exhaustive : celle-ci consiste à établir tous les plans d'exécution possibles et pour chacun d'eux on évalue son coût d'exécution sur base du modèle des coûts. On choisira le plan le moins coûteux. L'avantage de cette solution est d'obtenir un plan optimal mais le temps nécessaire pour trouver cette optimum peut être long.

B. Stratégie basée sur des heuristiques : Afin de diminuer le temps d'exécution de l'algorithme d'optimisation, on envisagera non plus une approche exhaustive mais une approche heuristique. Celle-ci ne fournira plus l'optimum mais le temps d'exécution de cette algorithme sera moindre. Les règles heuristiques définies varient d'un algorithme à l'autre. Le nombre d'arguments ou la taille d'une requête peuvent par exemple faire l'objet d'une règle heuristique.

C. Stratégie mixte : elle mélange les deux types de solutions précédentes. Dans un premier temps on utilise des règles heuristiques pour limiter le nombre de plans d'exécution alternatifs et ensuite on évalue le coût de chacun des plans d'exécution restant pour en déduire le moins coûteux.

Cette comparaison nous montre combien il faut **concilier la complexité de l'algorithme d'optimisation avec la qualité du résultat qu'il fournira**. On se contentera souvent d'une "sorte d'optimum" plutôt que de l'optimum étant donné le contexte dans lequel on se situe. On préférera en effet obtenir un plan d'exécution plus ou moins optimal mais en quelque secondes plutôt qu'obtenir un plan d'exécution optimal mais en plusieurs jours.

4.5.7. Effet de l'architecture physique sur l'optimisation globale

En effet deux facteurs architecturaux peuvent influencer considérablement cette étape du traitement:

1. Le **partage des tâches** de traitement des sous-requêtes entre le SGBDH et les systèmes supportants les SBD locaux. Ce facteur permet de travailler en parallèle et donc de minimiser le temps CPU.

2. La **topologie du réseau** qui permet la communication entre les SBD locaux et le SGBDH. Suivant les possibilités de communications entre systèmes locaux, on peut envisager de limiter les volumes de transferts vers le SGBDH en procédant à des étapes d'intégration partielle de données.

A. Le partage des tâches

Etant donné que chaque SBD local dispose de ses propres ressources, on peut envisager d'utiliser leurs ressources pour exécuter certaines tâches et ainsi permettre le travail en parallèle. On peut envisager différentes approches à ce niveau qui font varier le travail du SGBDH et ceux des systèmes locaux. Pour simplifier l'exposé de ces approches on distinguera les composants suivants : le **Gestionnaire Global(GG)** et les **Interfaces Locales(IL)**. Le gestionnaire global fait partie du SGBDH tandis que chacune des interfaces locales fait partie d'un système sur lequel est implanté le SBD local correspondant.

Quatre **approches** peuvent dès lors être adoptées pour la conception de l'optimisation globale. Elles se distinguent par le niveau de complexité des gestionnaire global et interfaces locales.

Première approche : Gestionnaire global complexe et pas d'interface locale

Le gestionnaire global est chargé de tout le traitement de la requête globale, depuis la décomposition jusqu'à la traduction dans chacun des formalismes locaux. Il doit exécuter ces différentes étapes pour chaque sous-requête et seul les accès aux BD sont réalisés en parallèle et par les SGBD locaux évidemment. Le GG sera également compétent pour la traduction et l'intégration des données. Sa charge est donc extrêmement élevé, le temps CPU également ainsi que les transferts entre SGBDH et SBD locaux car aucune étape intermédiaire permettant de limiter le temps CPU ou les transferts n'est possible dans de telles conditions. Cette solution offre l'avantage que l'on intervient nullement au niveau des systèmes locaux, en ce sens que l'on utilise pas leurs espaces de travail ni de stockage et que l'on alourdit pas la charge de leurs CPU vu que tout est réalisé au niveau global.

Seconde approche: Interfaces locales et gestionnaire global simples

Le gestionnaire global (GG) transforme les requêtes globales en un ensemble de requêtes locales les plus **petites** possibles. Les interfaces locales (IL) reçoivent ces multiples requêtes qu'ils traduisent et font exécuter. Ensuite, ils renvoient les résultats au gestionnaire global qui se charge de leur intégration. Cette technique est celle utilisée dans Multibase.
Le gestionnaire global assume donc une lourde charge et le volume de communication entre le

GG et les IL est élevé.

Troisième approche: Interfaces locales et gestionnaire global de complexité moyenne

Le gestionnaire global transforme les requêtes globales en un ensemble de requêtes locales les plus **larges** possibles (une requête pour chaque SBD local). Ainsi moins de requêtes sont envoyées aux IL et les résultats fournis au GG sont plus complets.
Le GG a donc une charge de travail plus réduite. De même le volume de communication entre GG et IL est moindre.

Quatrième approche: Interfaces locales et gestionnaire global complexes

Le GG génère des programmes efficaces qui impliquent la participation des IL dans l'optimisation globale. Des résultats partiels peuvent être envoyés au GG ou aux IL. Ces derniers supportent des fonctionnalités supplémentaires telles la suppression des duplicats, la gestion et fusion de résultats temporaires. Cette technique est celle adoptée par DDTS et Mermaid.
La communication entre GG et IL est encore plus réduite et la charge du GG moindre. Cette approche maximise le travail local, qui est exécuter en parallèle par les SBD locaux et minimise le temps de transfert entre composants.

B. La topologie du réseau de communication

Il faut en effet remarquer l'importance de la topologie du réseau reliant les différents SBD locaux et le SGBDH. En effet celle-ci va déterminer les possibilités de communication entre les systèmes supportant les SBD locaux, ce qui peut influencer notre stratégie d'optimisation. En effet, le fait de pouvoir réaliser des étapes d'intégration partielle de données au niveau des schémas composants peut limiter le volume de données à transférer vers le SGBDH. Rappelons les trois topologies classiques de réseau (figure 4.48): topologie BUS, topologie ETOILE, topologie ANNEAU.

On remarque que la topologie en étoile ne permet la communication entre deux systèmes locaux que via le système global, ce qui est loin d'être avantageux pour notre problème.
Les deux autres typologies permettent des communications plus aisées entre deux systèmes quelconques mais la topologie ANNEAU est nettement moins avantageuse que la topologie BUS. On choisira généralement et cela pour des raisons de facilité et d'efficacité de concevoir un réseau où chaque système local peut communiquer avec un autre système local quelconque, appelé "full-connected network".

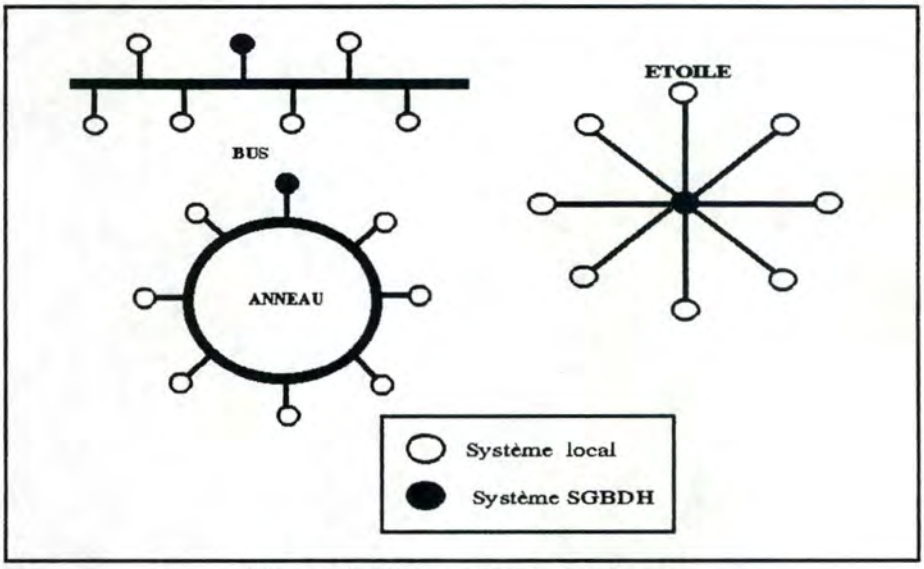


Figure 4.48: Topologie de réseau

4.5.8. La solution retenue pour l'optimisation

Des considérations dont il a été question précédemment on peut tirer les conclusions suivantes.
La définition d'un algorithme d'optimisation globale devrait prendre en compte les facteurs "CPU" et "communication", étant donné l'impossibilité de prendre en compte le facteur "temps d'accès BD" à ce niveau en raison de l'autonomie des SBD locaux. Mais nous sommes également dans l'impossibilité de définir un modèle de coût vu que les systèmes locaux sont dans l'impossibilité de nous fournir les données nécessaires à l'utilisation d'un tel modèle.
De ce fait la seule chose que l'on puisse faire pour optimiser les requêtes sont les suivantes:

- profiter au maximum du parallélisme pour traiter les sous-requêtes : cela dépend de l'architecture d'implantation du SBDH
- procéder à des semi-jointures pour limiter les volumes de transferts : cela dépend en autre de la topologie du réseau de communication
- traduire au mieux les requêtes dans les langages locaux pour profiter des avantages des SGBD locaux
- définir une phase d'optimisation locale si elle n'est pas implicitement réalisée par le SGBD local

Avant d'en terminer avec ce problème, on énoncera les problèmes relatifs à un type particulier de requête globale puisqu'il s'agit des transactions globales.

4.5.9. Les transactions globales

On se contentera de soulever le problème car il s'agit d'un problème dont l'origine est d'ordre technique et exceptionnelle. En effet une telle gestion n'est nécessaire que si des problèmes locaux (panne, crash disk, coupure de courant,...) ou de communication (rupture de câbles,...) se produisent. Jusqu'à présent nous faisions l'hypothèse irréaliste que "Tout va pour le mieux dans le meilleur des mondes" c'est-à-dire qu'aucun problème technique ne pouvait survenir.

Une transaction est une séquence d'opérations de consultations et de modifications que l'utilisateur considère comme une seule requête. Ainsi une transaction doit vérifier les trois propriétés suivantes:

- l'**atomicité** : une transaction sera exécutée complètement ou pas du tout
- l'**isolation**: l'exécution d'une transaction doit être isolée des autres transactions concurrentes
- la **durabilité** : l'exécution d'une transaction doit persister dans les BD si elle a été terminée

Garantir ces propriétés dans le contexte des BDH est difficile pour les raisons suivantes:

- les techniques de gestion des transactions varient d'un SGBD local à l'autre
- garantir les propriétés des transactions implique la restriction de l'autonomie des SBD locaux.

Il faut donc développer des techniques de gestion de concurrence, de gestion de commit adéquates au contexte analysé. [PU 91], [PERRIZO 91], [GLIGOR 86], [BREITBART 88] analysent ces problèmes.

CHAPITRE 5: ILLUSTRATION SYNTHETIQUE

Nous consacrerons ce chapitre à l'exposé d'une illustration qui constituera une synthèse des problèmes et solutions que nous avons abordés dans le chapitre 4. Ainsi, on définira une cas de bases de données hétérogènes et dans un premier temps on produira l'ensemble des éléments (schémas composants, schéma intégré, mappings,...) nécessaires à la gestion des ces BDH. Dans un second temps, on définira quelques requêtes globales dont on simulera les différentes étapes d'exécution.

Supposons deux bases de données dont les schémas et données sont présentés aux figures 5.1. et 5.2.
La base de données BD1 est une BD de type relationnelle à laquelle est associé le langage SQL.
La seconde (BD2) est de type réseau Codasyl avec le langage Codasyl DML
Nous avons choisi des BD dont les modèles sont exclusivement relationnel ou réseau Codasyl afin de permettre au lecteur de se référer aux transformations de la section 4.3.

Base de données 1(BD1)

COMPAGNIE - ASSURANCE

Nom-Cie	Catégorie	Capital
STRING[20]	*STRING[2]*	*INTEGER*
WINTERTHUR	A1	12
A.G.	B2	8
ROYAL BELGE	A2	10

Key(Compagnie Assurance)=Nom-Cie

CLIENT

Num-C	Nom-C	Prénom-C	Localité-C	Téléphone-C
STRING[6]	*STRING[20]*	*STRING[20]*	*STRING[25]*	*STRING[10] U NULL*
PO5874	*DUPONT*	*JEAN*	*NAMUR*	*081/228547*
TY8741	*DURANT*	*ROGER*	*LIEGE*	*041/583541*
KT9874	*KERKOVE*	*JAN*	*BRUXELLES*	*02/7513698*
GA0390	*COPPENS*	*MARIE*	*MOUSCRON*	*NULL*
TR9600	*DUFOUR*	*PIERRE*	*TOURNAI*	*069/239835*

Key(CLIENT)=Num-C

CONTRAT

Num-Contrat	Type	Cotisation Mens.	Num-Assuré	Nom-Compagnie
STRING[9]	*STRING[4]*	*INTEGER*	*STRING[6]*	*STRING[20]*
AZ2569852	*VIE*	*1100*	*PO5874*	*A.G.*
DE1298755	*FEU*	*1400*	*TR9600*	*WINTHERTUR*
PA6852257	*VEH*	*1300*	*TY8751*	*A.G.*
MO6895231	*FEU*	*1250*	*KT9874*	*ROYAL BELGE*
QA2665898	*VOL*	*1700*	*GA0390*	*WINTHERTUR*
UR5896521	*VEH*	*1200*	*TR9600*	*ROYAL BELGE*
PI5965386	*VIE*	*1900*	*KT9874*	*ROYAl BELGE*

Key(Contrat)=Num-contrat
Nom-Compagnie $\subset$ COMPAGNIE-ASSURANCE.Nom-Cie
Num-Assuré $\subset$ CLIENT.Num-C

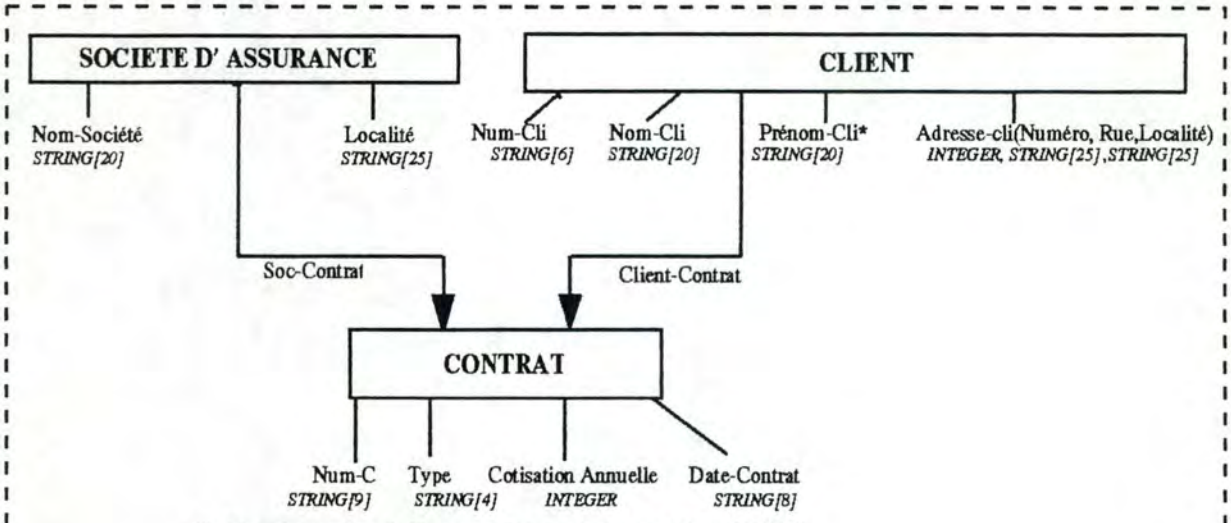
AGENT

Num-A	Nom-A	Prénom-A	Localité-A	Nom-Cie
STRING[5]	*STRING[20]*	*STRING[20]*	*STRING[25]*	*STRING[20]*
A2145	*DUBUISSON*	*PASCAL*	*TOURNAI*	*A.G.*
A6985	*DUMONT*	*MARIE*	*NAMUR*	*ROYAL BELGE*
A2452	*VERMEESH*	*MARC*	*BRUXELLES*	*WINTHERTUR*
A9652	*DUBOIS*	*PAUL*	*MOUSCRON*	*WINTHERTUR*

Key(Agent)=Num-A
Nom-Cie = COMPAGNIE-ASSURANCE.Nom-Cie

Figure 5.1: Schéma et données de la BD1

Base de données 2(BD2)



" **Tout contrat figure dans chacun des SET**"

Identifiant(Société d'Assurance)=Nom-Société

Identifiant(Client)=Num-Cli

Identifiant(Contrat) = Num-c

SOCIETE D'ASSURANCE

Nom-Société	Localité
WINTERTHUR *A.G.* *ROYAL BELGE*	*NAMUR* *BRUXELLES* *LIEGE*

CONTRAT

Num-C	Type	Cotisation Annuelle	Date-Contrat
AZ2569852	*VIE*	*13200*	*21121990*
DE1298755	*FEU*	*16800*	*14021985*
PA6852257	*VEH*	*15600*	*01061982*
MO6895231	*FEU*	*15000*	*02091956*
QA2665898	*VOL*	*20400*	*20091967*
UR5896521	*VEH*	*14400*	*11011969*
PI5965386	*VIE*	*22800*	*04081981*

CLIENT

Num-Cli	Nom-Cli	Prénom-Cli	Adresse-Cli(Numéro, Rue, Localité)
PO5874	*DUPONT*	*JEAN*	*(25, GODEFROID,NAMUR)*
TY8741	*DURANT*	*ROGER,PAUL*	*(24, SAINTE MARIE,LIEGE)*
KT9874	*KERKOVE*	*JAN*	*(28, SQUARE VERGOTE, BRUXELLES)*
GA0390	*COPPENS*	*MARIE, SOPHIE*	*(125, DE LA LOUPE,MOUSCRON)*
TR9600	*DUFOUR*	*PIERRE*	*(52, ROYALE,TOURNAI)*

Soc-Contrat: SOCIETE D'ASSURANCE-CONTRAT

SOCIETE D'ASSURANCE: Nom-Société	CONTRAT:Num-C
WINTERTHUR *A.G.* *ROYAL BELGE*	*DE1298755 , QA2665898* *AZ2569852 , PA6852257* *MO6895231 , UR5896521* *PI5965386*

Client-Contrat: CLIENT-CONTRAT

CLIENT:Num-Cli	CONTRAT: Num-c
PO5874	AZ2569852
TR9600	DE1298755 , UR5896521
TY8751	PA6852257
KT9874	MO6895231 , PI5965386
GA0390	QA2665898

Figure 5.2 :Schéma et données de la BD2

Conformément à notre architecture de solution (figure3.1), nous allons définir les ***schémas composants*** relatifs aux schémas de chacune de ces BD locales. Rappelons que ces schémas

composants sont l'expression dans un formalisme unique (le modèle canonique EA/E) des schémas locaux. Nous utiliserons les transformations de schémas exposées à la section 4.3.C.

Les figures 5.3. et 5.4 illustrent les schémas composants des schémas des figure 5.1. et 5.2. Ces schémas seont accompagnés des règles de mapping.

Schéma composant de BD1

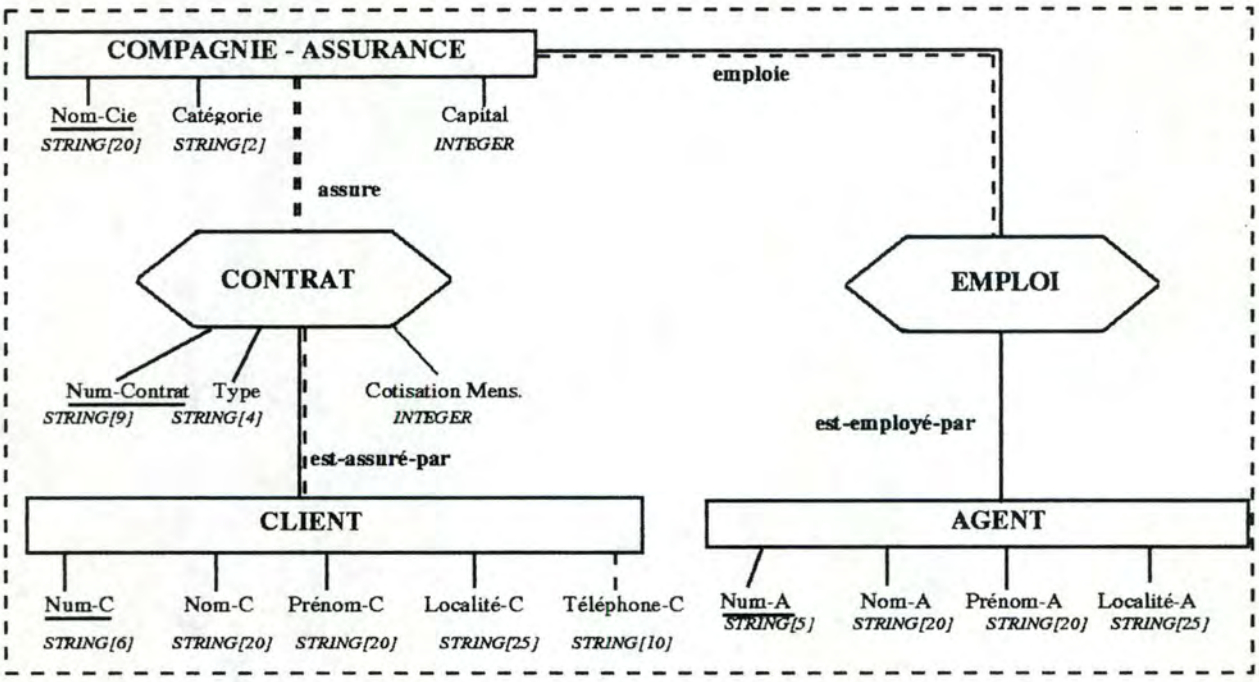


Figure 5.3.: Schéma composant de BD1

Règles de mapping Syntaxique pour la BD1

Compagnie-Assurance(Nom-Cie,Catégorie,Capital)
↵ Compagnie-Assurance(Nom-Cie,Catégorie,Capital)

Client(Num-C,Nom-C,Prénom-C,Localité-C,Téléphone-C)
↵ Client(Num-C,Nom-C,Prénom-C,Localité-C, Téléphone-C)

Agent(Num-A,Nom-A,Prénom-A,Localité-A) ↵ Agent(Num-A,Nom-A,Prénom-A,Localité-A)

Emploi(Agent: employé-par, Compagnie-Assurance: emploie) ↵ Agent(Num-A,Nom-Cie)

Contrat(Num-Contrat, Type, Cotisation Mens., Compagnie-Assurance, Client:assuré-par,

Compagnie-Assurance: assure) ↵ Contrat(Num-Contrat, Type, Cotisation Mens., Num-assuré, Nom-compagnie)

Compagnie-Assurance **emploie** Agent ↵ AGENT.Nom-Cie = Compagnie-Assurance.Nom-Cie
Agent **employé-par** Compagnie-Assurance ↵ AGENT.Nom-Cie=Compagnie-Assurance.Nom-Cie
Compagnie-Assurance **assure** Client ↵
Contrat.Nom-Compagnie **C** Compagnie-Assurance.Nom-Cie
Contrat. Nom-Assuré = Client.Num-C
Client **assuré par** Compagnie-Assurance ↵
Contrat. Nom-Compagnie **C** Compagnie-Assurance.Nom-Cie
Contrat. Nom-Assuré = Client.Num-C

Règles de mapping des occurences pour la BD 1

Compagnie-Assurance(Nom-Cie,Catégorie,Capital)
= Compagnie-Assurance(Nom-Cie,Catégorie,Capital)

Client(Num-C,Nom-C,Prénom-C,Localité-C)
= Client(Num-C,Nom-C,Prénom-C,Localité-C, Téléphone-C)

Client(Téléphone-C) = Π [Téléphone-C] Client **si** ≠ NULL
= ' ' **sinon**

Agent(Num-A,Nom-A,Prénom-A,Localité-A)=Π[Num-A,Nom-A,Prénom-A,Localité-A] Agent

Emploi(Agent: employé-par, Compagnie-Assurance: emploie) = Π [Num-A,Nom-Cie]Agent

Contrat(Num-Contrat, Type, Cotisation Mens., Compagnie-Assurance, Client:assuré-par, Compagnie-Assurance: assure) = Contrat(Num-Contrat, Type, Cotisation Mens., Num-assuré, Nom-compagnie)

Agent: (Compagnie-Assurance.Nom-Cie =X) **emploie** Agent = σ [Nom-Cie =X]Agent

Compagnie-Assurance: (Agent.Num-A=Y) **employé-par** Compagnie-Assurance = σ [Num-A=Y] Agent

Client: (Compagnie-Assurance.Nom-Cie=X) **assure** Client = Π [Num-Assuré] σ [Nom-Compagnie =X] Contrat

Compagnie-Assurance: (Contrat.Num-Assuré=Y) **assuré par** Compagnie-Assurance =Π [Nom-Cie] σ [Num-Assuré =Y] Contrat

Schéma composant de BD2

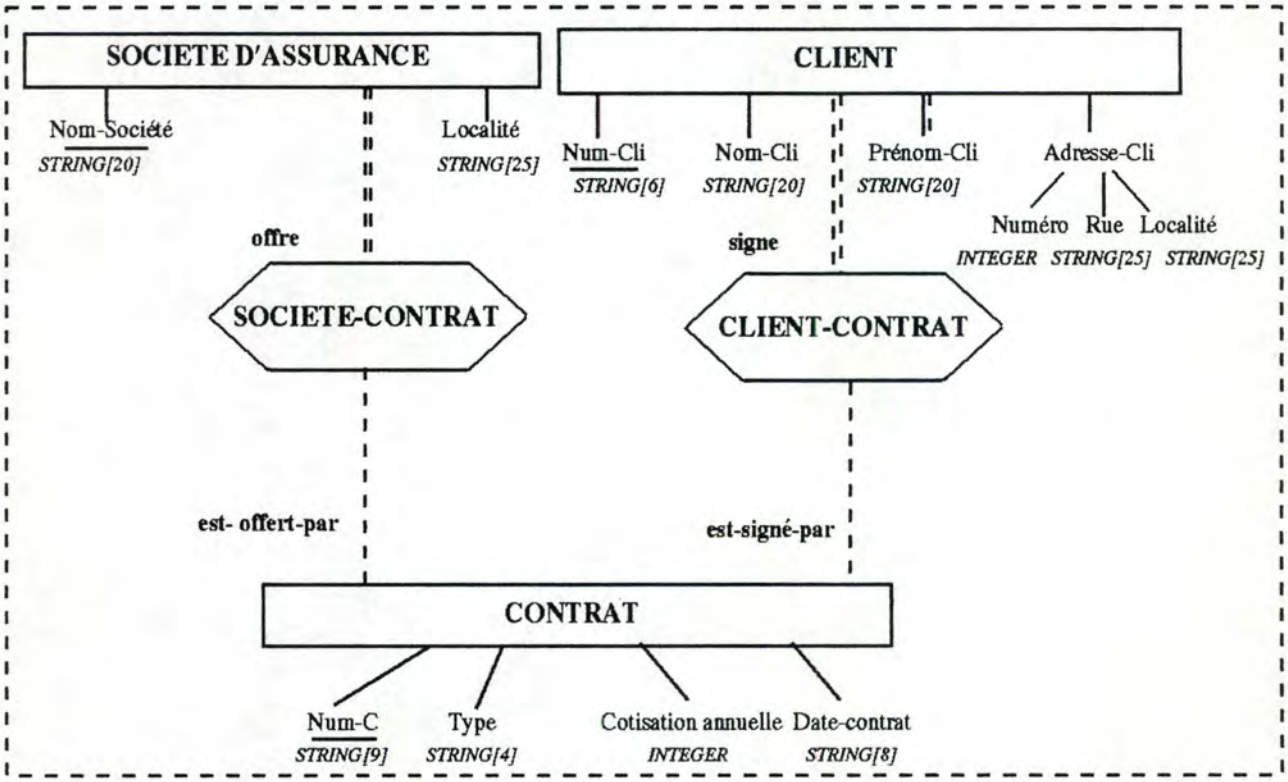


Figure 5.4.: Schéma composant de BD2

Règles de mapping syntaxique pour la BD 2

Société d'Assurance(Nom-Société, Localité) ↵ Société d'Assurance(Nom-Société, Localité)

Client(Num-cli,Nom-cli,Prénom-Cli,Adresse-Cli(Numéro,Rue,Localité)) ↵
Client(Num-cli,Nom-cli,Prénom-Cli,Adresse-Cli(Numéro,Rue,Localité))

Contrat(Num-C,type,Cotisation annuelle,Date-contrat) ↵
Contrat(Num-C,type,Cotisation Annuelle,Date-contrat)

Société-Contrat(Société D'Assurance: offre, Contrat:est-offert-par) ↵
Soc-Contrat(Société D'Assurance, Contrat)

Client-Contrat(Client:signe,Contrat:est-signé-par) ↵ Client-Contrat(Client,Contrat)

Société D'assurance **offre** Contrat ↵ SociétéD'assurance **Soc-Contrat** Contrat

Contrat **offert-par** Société D'assurance ↵ Contrat Soc-Contrat Société D'assurance

Client **signe** Contrat ↵ Client **Client-Contrat** Contrat

Contrat **signé-par** Client ↵ Contrat **Client-Contrat** Client

Mapping des occurences pour la BD2

Société d'Assurance(Nom-Société, Localité) = Société d'Assurance(Nom-Société, Localité)

Client(Num-cli,Nom-cli,Prénom-Cli,Adresse-Cli(Numéro,Rue,Localité)) =
Client(Num-cli,Nom-cli,Prénom-Cli,Adresse-Cli(Numéro,Rue,Localité))

Contrat(Num-C,type,Cotisation annuelle,Date-contrat) =
Contrat(Num-C,type,Cotisation Annuelle,Date-contrat)

Société-Contrat(Société D'Assurance: offre, Contrat:est-offert-par) = (OWNER,MEMBER) OF Soc-Contrat

Client-Contrat(Client:signe,Contrat:est-signé-par) = (OWNER,MEMBER) OF Client-Contrat

Contrat: Société D'assurance **offre** Contrat = MEMBER OF **Soc-Contrat**

SociétéD'Assurance: Contrat **offert-par** Société D'assurance = OWNER OF Soc-Contrat

Contrat: Client **signe** Contrat = MEMBER OF **Client-Contrat**

CLIENT: Contrat **signé-par** Client = OWNER OF **Client-Contrat**

Nous ferons abstraction de la définition des schémas exportés(sous-schémas des schémas composants) étant donné qu'il s'agit de l'étude d'un cas d'école.
Létape suivante pour la résolution de cet exemple de BDH consiste donc à **intégrer** les deux schémas composants. Nous allons donc, conformément à notre méthode (section4.4.6), commencer par définir les assertions de correspondances entre les schémas des figure 5.3 et 5.4.

Assertions de correspondances

Compagnie-Assurance ≡ Société D'Assurance WCA Nom-Cie = Nom-Société

Client ≡ Client WCA Num-C = Num-Cli,
Nom-C=Nom-Cli,
Prénom-C C Prénom-Cli,
Localité-C = Adresse-Cli.Localité

Contrat ≡ Contrat WCA Num-contrat=Num-C,
Type=Type,
Cotisation-Mens = 1/12*(Cotisation-Annuelle)

Compagnie-Assurance - Contrat = Société D'Assurance - Société-Contrat - Contrat

Client - Contrat ≡ Client - Client-Contrat - Contrat

Sur base de ces correspondances, nous obtenons le schéma intégré de la figure5.5.

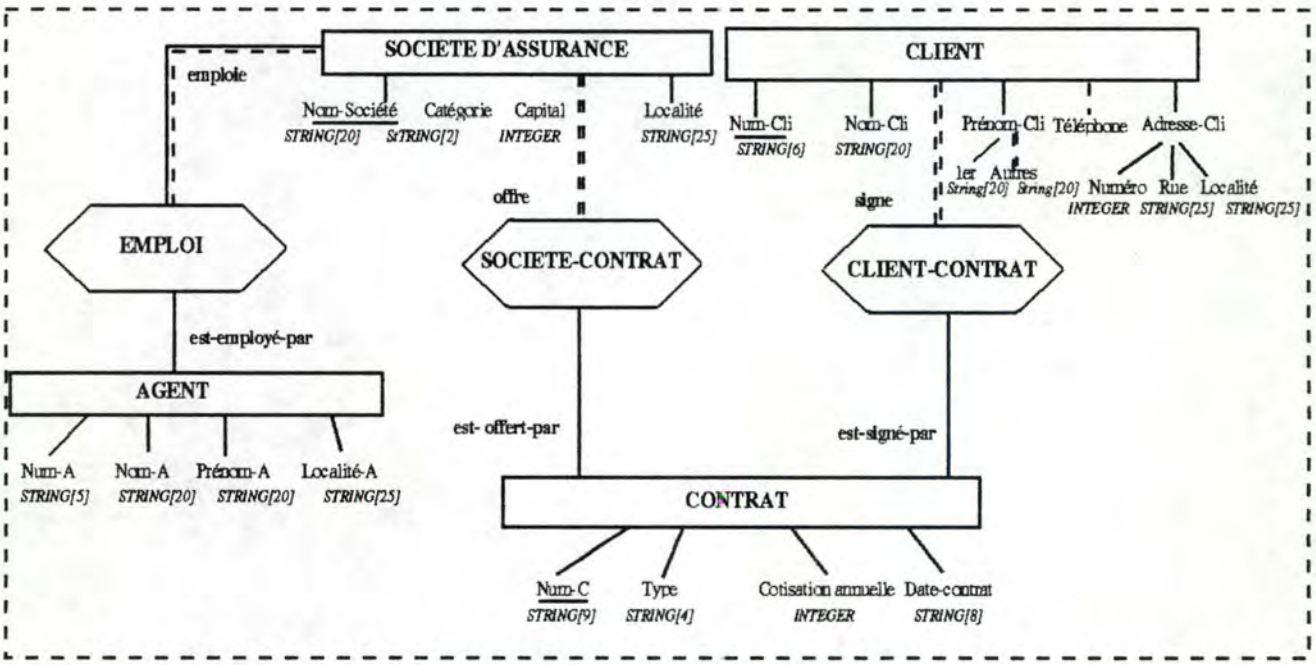


Figure 5.5. Schéma intégré des figures 5.3 et 5.4

On pourrait affiner le schéma intégré en transformant les deux TA SOCIETE-CONTRAT et CLIENT-CONTRAT et le TE CONTRAT en un seul TA CONTRAT. Nous ne le ferons pas pour la simple raison que dans ce cas-ci, les règles de mapping n'en seraient pas simplifiées. Il suffit d'examiner les schémas composants pour constater les ressemblances entre le premier ou le second schéma suivant qu'on réalise ou pas cette transformation. En réalité, cette transformation sera toutefois réalisée afin de rendre le schéma plus simple(Cfr structure de la figure 5.3).

Les règles de mapping entre le schéma intégré et les schémas composants sont les suivantes:

Règles de mapping syntaxique

Société D'Assurance ≡ BD1.Compagnie-Assurance WCA Nom-Société=Nom-Cie,
Catégorie=Catégorie,
Capital=Capital

Société D'Assurance ≡ BD2.Société D'Assurance WCA Nom-Société=Nom-Société,
Localité=Localité

Client ≡ BD1.Client WCA Num-Cli = Num-C,
Nom-Cli=Nom-C,
Prénom-Cli.1er=Prénom-C,
Adresse-Cli.Localité=Localité-C,
Téléphone-Cli=Téléphone-C

Client ≡ BD2.Client WCA Num-Cli = Num-Cli,
Nom-Cli=Nom-Cli,
Prénom-Cli(1er,Autres)=Prénom-Cli,
Adresse-Cli=Adresse-Cli,

Agent ≡ BD1. Agent WCA Num-A=Num-A,
Nom-A=Nom-A,
Prénom-A, Prénom-A,
Localité-A=Localité-A

Emploi ≡ BD1. Emploi

Société D'Assurance - Emploi - Agent ≡ BD1.Compagnie-Assurance - Emploi - Agent

Contrat ≡ BD1.Contrat WCA Num-C= Num-Contrat,
Type= Type,
Cotisation annuelle = 12 * Cotisation Mens.

Contrat ≡ BD2.Contrat WCA Num-C= Num-C,
Type= Type,
Cotisation annuelle = Cotisation annuelle ,
Date-Contrat = Date-Contrat

Société-Contrat ≡ BD2.Société-Contrat

Client-Contrat ≡ BD2.Client-Contrat

Société D'assurance- Société-Contrat - Contrat ≡ BD1. Compagnie-Assurance - Contrat

Client - Client-Contrat - Contrat ≡ BD1.Client - Contrat

Société D'assurance- Société-Contrat - Contrat ≡ BD2. Société D'assurance - Société-Contrat - Contrat

Client - Client-Contrat - Contrat ≡ BD2.Client - Client-Contrat - Contrat

Mapping des occurences

Agent = BD1. Agent
Emploi = BD1.Emploi
Société D'Assurance = Integrate-Join(BD1.Compagnie -Assurance, BD2.Société S'Assurance, Nom-Cie = Nom-Société)

Client = Integrate-Join (BD1.Client,BD2.Client, Num-C = Num-Cli,
Nom-C=Nom-Cli,
Prénom-C **C** Prénom-Cli,
Localité-C = Adresse-Cli.Localité)

Contrat = Integrate-Join(BD1.Contrat,BD2.Contrat,Num-contrat=Num-C,
Type=Type,
Cotisation-Mens = 1/12*(Cotisation-Annuelle)

Société-Contrat (Société D'Assurance, Contrat) = BD1.{ π [Nom-Cie , Contrat^{i}.Contrat.Num-Contrat] }Compagnie-Assurance/assure * (Contrat, Client/est-assuré-par)
avec i:1 à Card(Contrat)

PS: **{}** représente la notion d'ensemble de plusieurs occurrences

Société-Contrat (Société D'Assurance, Contrat) = BD2.Société-Contrat(Société D'Assurance, Contrat)

Client-Contrat (Client,Contrat) = BD1.{ π[Num-C,Contrat^{i}.Contrat.Num-Contrat] }
Client/est-assuré-par * (Contrat, Compagnie-Assurance/ssure)
avec i:1 à Card(Contrat)

Client-Contrat (Client,Contrat) = BD2.Client-Contrat(Client,Contrat)

Traitement de quelques requêtes adressées au schéma intégré

Nous traiterons successivement les requêtes suivantes :

Requête 1 :
"Donner les catégorie et localité de la société d'assurance de nom"WINTHERTUR" ainsi que le numéro, le nom et le prénom de leurs agents"

SELECT Catégorie, Localité, Num-A,Nom-A, Prénom-A
FROM SociétéD'Assurance, Agent
WHERE Nom-société = "WINTHERTUR" AND
Agent employé-par Société D'Assurance

Requête 2 :
"Donner les numéro et type des contrats ainsi que les numéro, adresse et téléphone des Clients dont la localité est "LIEGE" et qui ont passé un contrat avec la Société D'Assurance de nom"A.G."

SELECT Num-C, type, Num-Cli, Adresse-Cli,Téléphone
FROM Contrat, Client,Société D'Assurance
WHERE Contrat est-signé-par Client AND
Localité-Cli= "LIEGE"
AND Contrat est-offert-par Société D'Assurance AND
Nom-Cie = "A.G."

Requête 3 :
"Modification: déporter tous les contrats de type "VOL" signés par la société d'assurance WINTHERTUR vers la société A.G"

MODIFY Société-Contrat **ON** offre : Société D'Assurance **WITH** Nom-Société="A.G."
WHERE Contrat.type="VOL" AND Contrat est-offert-par Société D'Assurance AND Nom-Société="WINTHERTUR"

Les tableaux des pages suivantes tracent l'exécution de ces requêtes. Dans la marge gauche de ces tableaux, nous avons le symbole indiquant le traitement en cours:

DE: Décomposition de requête
TR:Transformation de requête
A: Accès aux données
TD: Transformation de données
I: Intégration des données

Traitement de la requête R1		
	SELECT Catégorie, Localité, Num-A,Nom-A, Prénom-A **FROM** SociétéD'Assurance, Agent **WHERE** Nom-société = "WINTHERTUR" AND Agent employé-par Société D'Assurance	
	Traitement relatif à BD1	**Traitement relatif à BD2**
DE	SELECT Nom-Cie, Catégorie, Num-A, Nom-A, Prénom-A FROM Compagnie-Assurance, Agent WHERE Agent employé-par Compagnie-Assurance AND Nom-Cie = "WINTHERTUR"	SELECT Nom-Société, Localité FROM Société D'Assurance WHERE Nom-Société="WINTHERTUR"
TR	SELECT Nom-Cie, Catégorie, Num-A, Nom-A, Prénom-A FROM Compagnie-Assurance, Agent WHERE Compagnie-Assurance.Nom-Cie = "WINTHERTUR" AND Agent.Nom-Cie = Compagnie-Assurance.Nom-Cie	MOVE "WINTHERTUR" TO Nom-Société OF Société D'Assurance. Error = 0. FIND Société D'Assurance RECORD. IF error =0 THEN GET Société D'Assurance; Nom-Société,Localité.
A	*(WINTHERTUR,A1,A2452,VERMEESH,MARC)* *(WINTHERTUR, A1,A9652,DUBOIS,PAUL)*	*(WINTHERTUR, NAMUR)*

T **D**	*WINTHERTUR, A1, A2452, VERMEESH, MARC* *A9652, DUBOIS, PAUL*	*WINTHERTUR, NAMUR*
I	*WINTHERTUR, A1, NAMUR, A2452, VERMEESH, MARC* *A9652, DUBOIS, PAUL*	

Traitement de la requête R2		
SELECT Num-C,Date-Contrat,Num-Cli,Adresse-Cli,Téléphone **FROM** Contrat, Client, Société D'Assurance **WHERE** Contrat est-signé-par Client AND Localité-Cli="LIEGE" AND Contrat offert-par Société D'Assurance AND Nom-société = "A.G."		
	Traitement relatif à BD1	**Traitement relatif à BD2**
DE	SELECT Num-Contrat, Num-C, Localité-C, Téléphone-C FROM Compagnie-Assurance, Client,Contrat WHERE Client est-assuré-par Compagnie-Assurance AND Nom-Cie = "A.G." AND Localité-C="LIEGE"	SELECT Num-C,Date-Contrat,Num-Cli,Adresse-Cli FROM Contrat, Client, Société D'Assurance WHERE Contrat est-signé-par Client AND Localité-Cli="LIEGE" AND Contrat offert-par Société D'Assurance AND Nom-Société = "A.G."
TR	SELECT Num-Contrat, Num-C, Localité-C, Téléphone-C FROM Client, Contrat WHERE Nom-Compagnie = "A.G." AND Num-Assuré IN SELECT Num-C FROM Client WHERE Localité-C = "LIEGE"	*Hyp: Clé d'accès sur **Localité** du record **Client**. Si cette hypothèse n'est pas vérifiée, il faudrait réaliser un accès séquentiel sur tous les record de **Client**.* MOVE "LIEGE" TO Localité OF Client. error=0. existe=0. FIND Client RECORD. If error=0 THEN PERFORM ***Traiter-contrats.*** If existe=1 THEN GET Client;Num-Cli, Adresse-Cli END-IF. PERFORM ***Traiter-Client-Suivant*** UNTIL error <> 0 END-IF. STOP

		Traiter-Contrats FIND FIRST Contrat RECORD OF Client-Contrat SET. PERFORM ***Traiter-Contrat-Suivant*** UNTIL error <> 0 **Traiter-Contrat-Suivant** FIND OWNER RECORD OF Soc-Contrat SET. IF Nom-Société OF Société D'Assurance ="A.G." move1 to existe GET Contrat;Num-C, Date-Contrat END-IF. FIND NEXT Contrat RECORD of Client-Contrat SET. **Traiter-Client-suivant** FIND NEXT DUPLICATE WITHIN Client RECORD . Perform ***Traiter-Contrats*** UNTIL error <> 0.
A	*(PA6852257, TY8741, LIEGE, 041/583541)*	Contrat :(*PA6852257, 01061982)* Client : (*TY8741, 24, (SAINTE MARIE, LIEGE))*
T **D**	*(PA6852257, TY8741, LIEGE, 041/583541)*	*PA6852257, 01061982, TY8741, (24, SAINTE MARIE, LIEGE)*
I	***PA6852257, 01061982, TY8741, (24, SAINTE MARIE, LIEGE), 041/583541***	

Traitement de la requête R3		
MODIFY Société-Contrat **ON** offre : Société D'Assurance **WITH** Nom-Société="A.G." **WHERE** Contrat.type="VOL" AND Contrat est-offert-par Société D'Assurance AND Nom-Société="WINTHERTUR"		
	Traitement relatif à BD1	**Traitement relatif à BD2**
D E	**MODIFY** Contrat ON assure:Compagnie-Assurance **WITH** Nom-Cie="A.G." **WHERE** Contrat.type = "VOL" AND Nom-Cie= "WINTHERTUR"	**MODIFY** Société-Contrat **ON** offre : Société D'Assurance **WITH** Nom-Société="A.G." **WHERE** Contrat.type="VOL" AND Contrat est-offert-par Société D'Assurance AND Nom-Société="WINTHERTUR"
T R	**UPDATE** Contrat **SET** Nom-Compagnie="A.G." **WHERE** Type="VOL"AND Nom-Compagnie="WINTHERTUR"	*Hyp:Nous émettons l'hypothèse qu'il existe une clé d'accès sur "Type" de "Contrat"* error-count = 0. MOVE "A.G." TO Nom-Société OF Société D'Assurance. FIND Société D'Assurance RECORD. MOVE currency-Status for record Société D'Assurance to Soc-Temp1. MOVE "WINTHERTUR " TO Nom-Société OF Société D'Assurance. FIND Société D'Assurance RECORD. MOVE currency-Status for record Société D'Assurance to Soc-Temp2. Move "VOL" TO Type OF Contrat. Perform ***Traiter-Contrat*** UNTIL error-count <>0. STOP

Traiter-Contrat

```
FIND Contrat VIA CURRENT OF Soc-Contrat SET USING Type;
IF error-count = 0
THEN
    REMOVE Contrat FROM Soc-Contrat
    FIND Société D'Assurance using Soc-Temp1
                                suppress run-unit currency updates
    INSERT Contrat INTO Soc-Contrat
    FIND Société D'Assurance using Soc-Temp2
END-IF.
```

CHAPITRE 5: CONCLUSIONS

Par l'analyse du problème des bases de données hétérogènes (BDH), nous avons eu l'occasion de mettre sur le tapis des problèmes fondamentaux relatifs aux bases de données. Nous pouvons citer les problèmes de transformations (de schémas, de données et de requêtes), d'intégration (de schémas et de données), de décomposition et d'optimisation de requêtes. Mais nous pouvons également rappeler le problème du choix du modèle et langage canonique qui a fait et continue à faire l'objet de longues discussions. Nous avons abordé chaque problème en donnant un exposé critique des diverses solutions possibles afin d'en déduire une solution appropriée au contexte des bases de données hétérogènes.

Etant donné la complexité et l'étendue du problème des BDH, nous n'avions ni pour objectif ni pour prétention d'y apporter une solution complète "clé en main". Notre préférence a été portée sur un exposé complet du problème en spécifiant la nature et les origines du problème, les philosophies de résolution, les architectures globales possibles, les différents problèmes sous-jacents et leurs solutions.

Pour chacun des problèmes traités, nous pouvons retenir les idées suivantes.

Au sujet de l'architecture globale de solution, nous avons opté pour celle dénommée "architecture fédérée fortement couplée mono-schéma". Ce choix a été justifié mais notons toutefois que cette justification faisait intervenir des facteurs qui peuvent varier en fonction des souhaits attendus d'un SGBDH.

A propos de choix du modèle canonique, nous avons exprimé notre souhait d'avoir un modèle de type entité-association étendu avec quelques extensions et accompagné d'un langage assertionnel de type calcul.

Concernant les transformations de schémas, nous avons exposé quelques règles de transformations pour les modèles relationnel et Codasyl. Afin de faire face à une plus grande variété de systèmes de bases de données, il faudrait développer les règles de transformations pour d'autres modèles de données. Pour les transformations de requêtes, nous nous sommes contentés d'en exposer les principes, ces derniers variant en fonction des types des langages sources et cibles.

Au sujet de l'intégration de schémas, notre préférence s'est tournée vers une méthode déclarative semi-automatique offrant un modèle de description de correspondances hétérogènes. A l'avenir, on pourrait envisager de définir des correspondances entre des modèles différents afin d'éliminer la phase de transformation de schémas. Mais il ne s'agit là que d'une approche qui nécessite encore réflexion.

Concernant le traitement des requêtes globales, nous avons signalé l'importance de la phase d'optimisation pour offrir des temps de réponse acceptables. La solution que nous avons

proposée était, pour des raisons de respect d'autonomie des systèmes de BD locaux, essentiellement liée à l'architecture d'implantation du SBDH.

Pour terminer, nous avons proposé une illustration synthétique afin de mettre en évidence l'ensemble des problèmes et les solutions apportées. Nous avons également pu montrer l'enchaînement des étapes qui mènent à la définition d'un SBDH et les différentes phases à accomplir pour l'exécution d'une requête globale. Par cette illustration, nous avons pu comprendre l'importance des règles de mapping pour traiter les requêtes globales et percevoir que des efforts sont encore à réaliser concernant la définition de ces règles, que ce soit au niveau de la transformation ou de l'intégration des schémas.

Bibliographie

[AHMED 91]: AHMED, DE SMEDT, DU & al., " *The PEGASUS heterogenous multidatabase systems*", in Computer, december 1991

[BATINI 86] : BATINI, LENZERINI, NAVATHE, " *A comparative analysis of methodology for database schema integration* ", in ACM computing surveys, vol. 18, n°4, december 1986

[BREITBART 88]: BREITBART, SILBERSCHATZ, "*Multidatabases updates issues*", in proceedings of the ACM Sigmod conference, june 1988, p.135-142

[BREITBART 90] : BREITBART, " *Multidatabases interoperability* ", in Sigmod record, vol 19, n°3, september 1990

[BRZEZINSKI 84]: BRZEZINSKI, GETTA, RYBNIK, STEPNIENSKI, " *UNIBASE: an integrated access to databases*", in Proceedings of the VLDB, 1984

[CARDENAS 87]: CARDENAS, " *Heterogeneous distributed databases management : the HD-DBMS*", in proceedings of the IEEE, vol 75, n°5, may 1987

[CASTELLANOS 91]: CASTELLANOS, SALTOR, " *Semantic enrichment of database schemas : an object-oriented approach* ", in proceedings of the workshop on interoperability in multidatabases systems, Kyoto 1991

[COLLET 91]: COLLET, HULUS, WEI MIN SHEN, " *Ressource integration using a large knowledge base in CARNOT*", in Computer, december 1991

[DAYAL 83] : DAYAL, " *Processing queries over generalization hierarchies in a multidatabase system*", in proceedings of the VLDB, 1983

[DAYAL 84] : DAYAL, H-Y WANG, " *View definition and generalization fordatabase integration in a multidatabase system* ", in IEEE Transaction on software Engineering, Vol SE-10, n°6, nov 1984

[DEMICHIEL 89]: DEMICHIEL, " *Resolving database incompatibility: an approach to performing relationnal operation over mismatched domains*" in IEEE transactions on knowledge and data engineering, 1, 4, 1989

[DEMURJIAN88]: DEMURJIAN, David K. HSIAO, " *Towards a better understanding of data models through the multilingual database system* ", in IEEE transactions on software engineering, vol 14, n° 7, july 1988

[FRANKHAUSER 88]: FRANKHAUSER, E.J. NEUHOLD, W. LITWIN 1988, *"Global View definition and Multidatabase languages- two approach to database integration"*, in Research into Networks and Distributed Application: proceedings of the European Teleinformatics Conference on Research into Networks and Distributed Application, Amsterdam, 1988

[GANDOPADHYAY 91] : GANDOPADHYAY, T. BARSALOU, " *The semantic equivalence of heterogeneous representation in multimodel, multidatabase systems*", in Sigmod record vol 20, n°4, december 1991

[GARCIA 91] : GARCIA, SOLACO, SALTOR, " *Discriminated operations on interoperable databases* ", in proceedings of the workshop on interoperability in

multidatabases systems, Kyoto 1991

[GLIGOR 86] : GLIGOR, POPESCU, " *Transaction management in distributed heterogeneous databases management system* ", in Information Systems, vol 11, n°4, 1986

[HAINAUT 90a] : HAINAUT, " *Entity-relationship models: formal specification and comparaison*", in proceedings of 9th International Conference on E-R approach, Lausanne, Octobre 1990

[HAINAUT 90b] : HAINAUT, " *The transformation toolkit of Tramis: analysis of pragmatic E-R transformations*", Researh report, University of Namur, 1990

[HAINAUT 91a] : HAINAUT," *Conception de bases de données: matières approfondies*", cours dispensé aux étudiants de seconde licence en informatique aux FUNDP, Institut d'Informatique, Namur, 1991

[HAINAUT 91b] : HAINAUT, "E *ntity-generating schema transformations for entity-relationship models*", in Proceedings of the 10th International Conference on Entity-Relationship Approach", San Mateo(CA), October 1991

[HEUER 90]: Heuer, Fuchs, Wiebking "OSCAR: *An object oriented database system with a nested relational network* ", in proceedings of 9 th International Conference on E-R approach, Lausanne, Octobre 1990

[HSIAO 89]: HSIAO, M.N. KAMEL, " *Heterogeneous databases: proliferations, issues and solutions* ", in IEEE Transaction on Knowledge and Data Engineering, vol 1, 1, March 1989

[KAUL 90] : KAUL, K. DROSTEN, E.J. NEUHOLD", *ViewSystem : Integrating Heterogeneous Information Bases by Object-Oriented Views* ", in proceedings of the 6th International Conference on data engineering, Los Angeles, february 1990

[KIM 91] : KIM, SEO, " *Classifying schematic and data heterogeneity in multidatabases systems*", in Computer, december 1991

[LITWIN 86] : LITWIN, ABDELLATIF, " *Multidatabase interoperability* ", in Computer, december 1986

[LITWIN 90] : LITWIN, L. MARK, N. ROUSSOPOULOS, " *Interoperability of multiple autonomous databases*", in ACM computing surveys, vol 22, n°3, sept 1990

[MARINOS 91] : MARINOS, PAPAZOGLOU, CHRISTODOULAKIS, " *Data integration methodology for an office environment* ", in Computer Systems, Science and Engineering, vol 6, n°3, july 1991

[MOTRO 87]: MOTRO, " *Superviews : Virtual Integration of multiple databases* ", in IEEE Transaction on software Engineering, Vol SE-13, n°7, july 1987

[OZSU 91] : OZSU, VALDURIEZ, " *Distributed databases systems, where are we now?*", in IEEE Computer, vol 24, n°8, août 1991

[PARENT 87] : PARENT, " *A model and an algebra for entity-relation type databases*", in Technology and Science of Informatics, vol 6, N°8, 1987

[PARENT 89] : PARENT,SPACCAPIETRA, " *About entities, complex objects end objects oriented data-models*", in Proceedings of an IFIP WC 8.1 Working Conference,

Namur, Oct. 18-20, 1989

[PERRIZO 91] : PERRIZO, J. RAJKUMAN, P. RAIN, " *HYDRO: a heterogeneous distributed database system* ",in proceedings of the 1991 ACM Sigmod International Conference on management of data, Denver, Colorado, may 1991

[RAM 91] : RAM, " *Heterogeneous distributed database systems* ", in Computer, december 1991

[REDDY 89] : REDDY, B.E. PRASAD, P.G. REDDY, " *A model for resolving incompatibilities and inconsistencies in integrating heterogeneous databases* ", in Management of data, W. Praskad Ed., MacGraw Hill, 1989

[ROLLAND 90] : ROLLAND, SOUVEYET, " *An object oriented framework for information systems*", COM AD 1990

[ROLLAND 92]: ROLLAND, BRUNET, "Object Database Design", in Proceeding of the European Conference on Engineering System Design ans Analysis, Istanbul,1992

[RUSINKIEWICZ 91]: RUSINKIEWICZ, "*Specifying interdatabase dependencies in a multidatabase environment*", in Computer, december 1991

[SALTOR 91]: SALTOR, CASTELLANOS, GARCIA-SOLACO " *Suitability of data models as canonical models for federated databases* ", in Sigmod Record, december 1991

[SHETH 90]: SHETH, J.A. LARSON, " *Federated databases systems for managing distributed, heterogeneous and autonomous databases* ", in ACM computing surveys, vol 22, n°3, september 1990

[SOPARKAR 91] : SOPARKAR, KORTH, SILBERSCHATZ, " *Failure resiliant transaction management in multidatabases*", in Computer, december 1991

[SPACCAPIETRA 90] , SPACCAPIETRA, PARENT, DUPONT, " *Automating Hetrogeneous Schema Integration* ", Rapport de recherche, Laboratoire de bases de donnnées, EPFL, Lausanne, 1990

[SPACCAPIETRA 91], SPACCAPIETRA, PARENT, " *Conflicts and correspondance assertions in interoperable databases*", in Sigmod Record, december 1991

[TEMPLETON 87]: TEMPLETON, BRILL, " *MERMAID : a front-end to distributed heterogeneous databases*", in proceedings of the IEEE, vol 75, n°5, may 1987

[THOMAS 90] : THOMAS, THOMPSON, al., " *Heterogeneous distributed database systems for production use*", in ACM computing surveys, vol 22, n°3, september 1990

[VENTRONE 91] : VENTRONE, HEILER, " *Semantic heterogeneity as a result of domain evolution*", in Sigmod record, vol 20, n°4, december 1991

[WEISHAR 91]: WEISHAR, KERSCHLEY, " *Data/knowledge packets as a mean of supporting semantic heterogeneity in multidatabase systems*", in Sigmod record, vol 20, n°4, december 1991

ERRATA

Les pages 101 et 108 manquent(erreur de numérotation des pages)

ANNEXES

ANNEXE 1 : Algèbre entité-association

Nous reprenons dans cette annexe quelques opérateurs algébriques définis sur le modèle entité-association et repris pour la plupart dans [PARENT 89].

Jointure

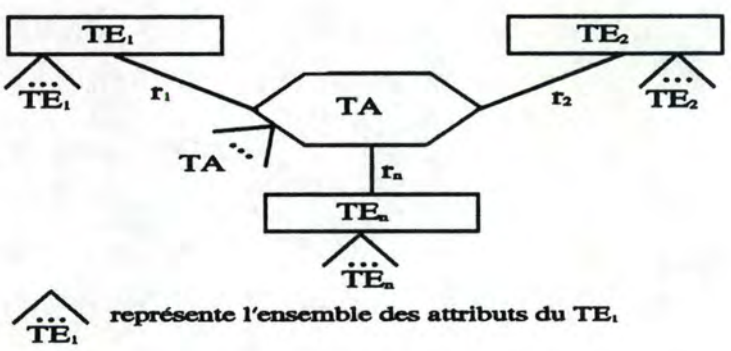
Soient deux types d'entités $TE_1(A_1,A_2,..,A_n)$ et $TE_2(A_1,B_1,B_2,..B_p)$ ayant un attribut en commun A_1. La jointure de TE_1 et TE_2 sur l'attribut A_1, consiste à fusionner les occurences de ces deux types d'entités en un seul type d'entité TE $(A_1,A_2,...,A_n,B_1,...,B_p)$. Ainsi, une occurence de TE est crée par composition d'une occurence de TE_1 et une occurence de TE_2 ayant même valeur pour l'attribut A_1.

On notera cet opération comme suit : TE = TE1 ⊗TE2 $[A_1]$

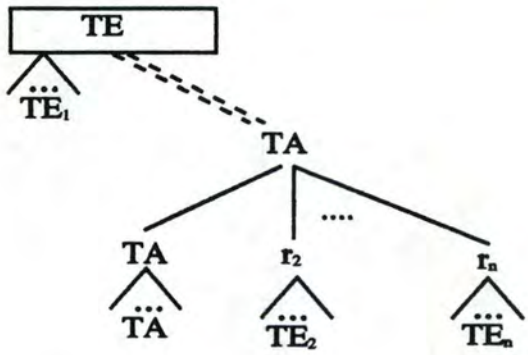
Jointure d'association (*)

Cette opération algébrique consiste à ramener sous un seul TE toutes les informations stockées sous plusieurs TE reliés par un TA.

Soit TA un type d'association entre les TE_1, TE_2,..., TE_n :



Le r-join de TE_1 via TA, noté TE = TE_1 / r_1 * (TA,TE_2 / r_2,...,TE_n / r_n) définit un nouveau type d'entité TE et ses attributs, dont la structure générale est la suivante:

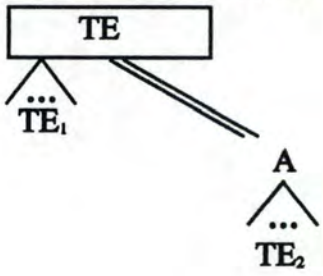


La connectivité du rôle joué par TE_1 détermine la cardinalité de l'attribut TA

Produit (X)

Un tel opérateur permet de connecter deux TE pour lesquels il n'existe pas de connexion. Ainsi chaque occurence du premier TE est "associée" à chaque occurence du second TE.
Soient deux TE TE_1 et TE_2 tel que A n'est pas le nom d'un attribut attaché directement à TE_1.

L'expression $TE = TE_1 \times (A : TE_2)$ définit un type d'entités TE et ses attributs tels que la structure en est la suivante :



Sélection(σ)

Cet opérateur crée un nouveau TE qui peut être vu comme un sous-type du type d'entité opérande.
$TE = \sigma\ [p]\ TE_1$ exprime la sélection sur base du prédicat p des occurences du type d'entité TE_1
Les occurences de TE sont donc le sous-ensemble des occurences de TE_1 qui vérifient p.
Un prédicat p peut être défini par :

- deux opérateurs logiques généralisés qui étendent les quantifieurs ∃ et ∀
- notation indicée pour associer les opérateurs logiques généralisés à la valeur désirée d'un attribut
- fonction de cardinalité qui permet de compter le nombre de valeurs d'un attribut(Card(*attribut*)=...)

Projection (π)

Cet opérateur permet de garder sous un nouveau TE seulement un sous-ensemble des attributs d'un TE opérande.
$TE = \pi [S] TE_1$ exprime la projection de TE_1 sur les attributs S

Réduction (ρ)

Cet opérateur permet d'éliminer les valeurs d'attribus non désirées dans les occurences existantes.

Soit TE_1 un type d'entités et **a** le chemin(A_n,...,A_1) entre TE_1 et l'attribut A_1 et **p** le prédicat définis sur les valeurs de A_1 pour une valeur du parent de A_1.
Les expressions

(1) $TE = \rho [* a // p] TE_1$
(2) $TE = \rho [(1) a // p] TE_1$ définissent les réductions de TE_1 sur base de l'attribut A

(1) représente le même ensemble d'occurences que TE_1 excepté que les valeurs de l'attributs A ne sont présentées que si elles vérifient p
(2) Représente les mêmes occurences que (1) en prenant soin d'éliminer les duplicats.

Union ($\cup$)

Définie sur des TE de même structures fournit l'union des occurences des deux TE
$TE = TE_1 \cup TE_2$

Différence (-)

Définie sur des TE de même structures fournit la différence des occurences de deux TE
$TE = TE_1 - TE_2$

Simplification(Σ)

Cet opérateur permet d'éliminer un niveau inutile dans une structure attribut. Un niveau est devenu inutile dès le moment où un attribut composé **a** n'a plus qu'un seul composant **b**. On peut dès lors supprimer ce composant **b**.
La notation utilisée est la suivante $E = \Sigma[a] E1$

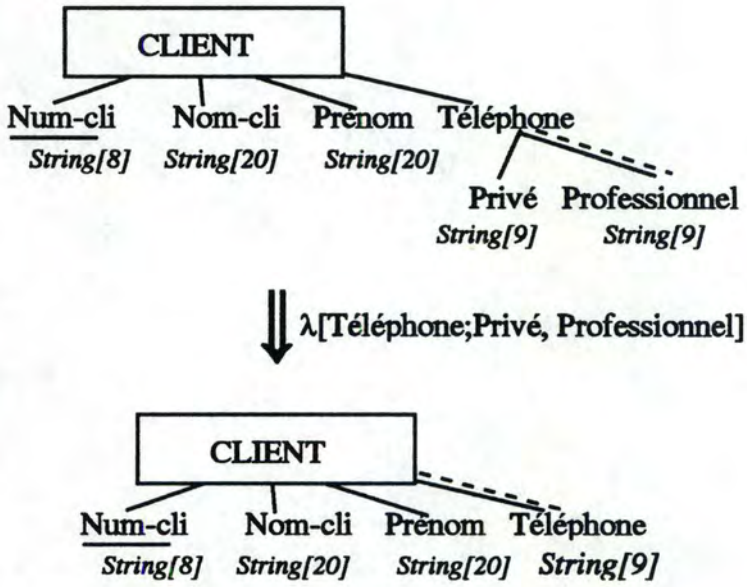
Jointure de spécialisation-Généralisation (Δ)

Cet opérateur est défini entre deux TE liés par une relation de généralisation : E2 is-a E1.

L'opération E = E1 Δ E2 crée un nouveau TE E reprenant les attributs de E1 et ceux de E2; ainsi E est constitué de l'ensemble des occurrences de E2 complétées par les attributs des occurrences de E1 auxquelles sont liés les occurrences de E2 par le lien is-a.

Fusion de composants d'un agrégat (λ)

Soient un TE avec un attribut composé A de cardinalité maximale 1 et composé des attributs $A_1,...,A_n$ dont les domaines sont identiques. L'opérateur de fusion consiste créer un attrribut répétitif A de domaine identique à ceux des A_i et atomique; cet attribut reprend l'ensemble des occurences des composants de l'attribut agrégat.
E = λ [A;A1,..,An] TE



$i^{ème}$ d'un attribut multivalué

Soit a, un attribut multivalué facultatif
a^i désigne la $i^{ème}$ valeur de cette attribut avec i: 0 ... card(a), avec a^0= ' '
Soit a, un attribut multivalué obligatoire
a^i désigne la $i^{ème}$ valeur de cette attribut avec i: 1 ... card(a)

Rename(α)

E =α [A:B] E1 opération qui définit un E identique à E1 excepté que le nom de l'attribut A de E1 est devenu B dans E

Integrate-join (opérateur d'intégration)

RAPPEL :
WCA signifie With Corresponding Attrribut et permet de définir les correspondances entre les attributs d'éléments ne correspondances.
attcor: exprime une des 4 correspondances possibles(égalité, inclusion, intersection, disjonction) entre deux attributs.

Soient E1 et E2 deux éléments (TE,TA,attributs composés) des schémas S1 et S2 respectivement et, $(A_{11},...,A_{1j},B_1,..,B_k)$ et ($A_{21},...,A_{2j},C_1,...,C_k$) les attributs respectifs de E1 et E2.

Soit l'assertion de correspondance:

$$E1 \equiv E2 \text{ WCA } attcor_1(A_{11},A_{21}), attcor_2(A_{12},A_{22}),..., attcor_j(A_{1j},A_{2j})$$

Soit $attcor_1(A_{11},A_{21})$ spécifie la correspondance(1-1) entre les identifiants de E1 et E2.

ALORS l'opérateur

$$\textbf{integrate-join(E1,E2, } \textbf{attcor}_1(A_{11},A_{21}), \textbf{attcor}_2(A_{12},A_{22}),..., \textbf{attcor}_j(A_{1j},A_{2j}))$$

crée un nouveau TE E défini comme suit:

- **structure de E** = Union des attributs de E1 et E2 définis comme suit :

  - un attribut B'_i pour chaque B_i de E1 sans correspondant dans E2, avec le domaine et cardinalités de B_i

  - un attribut C'_i pour chaque C_i de E2 sans correspondant dans E1, avec le domaine et cardinalités de C_i

  - un attribut A'i pour chaque $attcor_i(A_{1i},A_{2i})$ tel que $A_{1i} = A_{2i}$ ou $A_{1i} = f(A_{2i})$, avec le domaine et les cardinalités de A_{1i}

  - un agrégat $A'i(A_{1i},A_{2i}\text{-}A_{1i})$ pour chaque $attcor_i(A_{1i},A_{2i})$ tel que $A_{2i} \supset A_{1i}$

  - un agrégat $A'i(A_{1i}\text{-}A_{2i}, A_{2i}\text{-}A_{1i}, A_{1i} \cap A_{2i})$ pour chaque $attcor_i(A_{1i},A_{2i})$ tel que $A_{1i} \cap A_{2i}$

  - un agrégat $A'i(A_{1i},A_{2i})$ pour chaque $attcor_i(A_{1i},A_{2i})$ tel que $A_{1i} \neq A_{2i}$

  Pour les cardinalités des agrégats, nous renvoyons le lecteur à l'annexe 2 consacrée aux règles d'intégration.

- **population de E** = contient une occurence e pour chaque objet du monde réel du RWS de E1 et E2. La valeur de e est définie comme la fusion des valeurs des occurences e_1 de E1 et e_2 de E2 décrivant cet objet du monde réel et qui sont liées par la correpondance $attcor_1(A_{11},A_{21})$. Ainsi,

  - pour chaque attribut B'_i : $e.B'_i = e_1.B_i$

  - pour chaque attribut C'_i: $e.C'_i = e_2.C_i$

  - pour chaque attribut A'i :

si attcor(A_{1i},A_{2i}) est $A_{1i} = A_{2i}$ (CP: $A_{1i} = f(A_{2i})$) **alors** $e.A'_i = e_1.A_{1i}$ **ou*** $e_2.A_{2i}$
(CP: $e.A'_i = e_1.A_{1i}$ **ou*** $f(e_2.A_{2i})$)

si attcor(A_{1i},A_{2i}) est $A_{2i} \supset A_{1i}$ **alors** $e.A'_i.A_{1i} = e_1.A_{1i}$
$e.A'_i.A_{2i}-A_{1i} = e_2.A_{2i} - e_1.A_{1i}$

si attcor(A_{1i},A_{2i}) est $A_{1i} \cap A_{2i}$ **alors** $e.A'_i.A_{1i}-A_{2i} = e_1.A_{1i} - e_2.A_{2i}$
$e.A'_i.A_{2i}-A_{1i} = e_2.A_{2i} - e_1.A_{1i}$
$e.A'_i.A_{2i} \cap A_{1i} = e_1.A_{1i}$ si $e_1.A_{1i} = e_2.A_{2i}$
si attcor(A_{1i},A_{2i}) est $A_{1i} \neq A_{2i}$ **alors** $e.A'_i.A_{1i} = e_1.A_{1i}$
$e.A'_i.A_{2i} = e_2.A_{2i}$

X = A ou* B *signifie que X prend indifféremment la valeur A ou B et qu'en cas de conflit de valeur, il faut spécifier une règle de préférence (Cfr. intégration de données 4.4.7)*

ANNEXE 2 : Règles d'intégration

Pour définir ces quelques règles d'intégration, on s'est basé sur [SPACCAPIETRA 91]. La liste proposée n'est pas exhaustive. On s'est limité au problème des correspondances d'équivalence entre éléments excepté pour les attributs.

[1]. Intégration de deux attributs atomiques en correspondance de deux éléments (TE, TA ou attribut composé)en correspondance

PS: Concernant le mapping entre les attributs des occurences de TE,TA,..on évitera de noter l'occurrence de chaque attribut pour alléger l'écriture.

Soient E1 et E2 deux éléments en correspondance de deux schémas S1 et S2.
Soient A_1 et A_2 deux attributs atomiques de E1 et E2 respectivement.

Soit l'assertion de correspondance E1 ≡ E2 WCA attcor(A_1,A_2)

Si attcor(A_1, A_2) est :

- **$A_1 = A_2$**, l'intégration fournit le résultat suivant:

  Attribut atomique de nom A', de domaine dom(A_1), de cardinalités Card(A') = Card(A_1)
  Mapping Syntaxique: A'=S1.A_1 ; A'= S2.A_2
  S1.A_1 = A' ; S2.A_2 = A'
  Mapping des occurences: A_1≡S1.A_1 **ou*** S2.A_2 [au lieu de e.A' = S1.e_1.A_1 **ou*** S2.e_2.A_2]
  S1.A_1 = A' ; S2.A_2 = A'

- **$A_2 = f(A_1)$**, l'intégration fournit le résultat suivant:

  Attribut atomique de nom A', de domaine dom(A_1), de cardinalités Card(A') = Card(A_1)
  Mapping Syntaxique: A'=S1.A_1 ; A'=S2.A_2
  S1.A_1 = A' ; S2.A_2 = A'
  Mapping des occurences: A'. S1.A_1 **ou*** S2.f(A_2)
  S1.A_1 = A' ; S2.A_2 = f^{-1}.A'
  avec f, f^{-1} fonctions ou tables de conversion définies
  par le responsable de l'intégration

  X = A ou* B *signifie que X prend indifféremment la valeur A ou B et qu'en cas de conflit de valeur, il faut spécifier une règle de préférence (Cfr. intégration de données 4.4.7)*

- **$A_2 \supset A_1$**, l'intégration fournit le résultat suivant:

  Attribut composé de nom A' et de composants nommés A_1 et A_2-A_1.
  Les cardinalités de A' sont identiques à celles de S1.A_2; celles de A'.A_2-A_1 sont (0,N); celles de A.A_1 sont identiques à celles de A_1.
  Mapping syntaxique:A' = S2.A_2 ; A'.A_1=S1.A_1 ; A'.A_2-A_1 = S1.A_1,S2.A_2
  S1.A_1 = A'.A_1 ; S2.A_2 = A'
  Mapping des occurences: A' = S2.A_2 ; A'.A_1=S1.A_1; A'.A_2-A_1 = S2.A_2 - S1.A_1
  S1.A_1 = A'.A_1 ; S2.A_2 = A'

- **$A_1 \cap A_2$**, l'intégration fournit le résultat suivant:

  Attribut composé de nom A' et de composants nommés A_1-A_2, A_2-A_1, $A_1 \cap A_2$
  Les cardinalités de A' sont (i,1) avec i=0 ou 1 en fonction de la cardinalité minimale de A_1 et A_2. Celles de A'.A_1-A_2 sont (0,card-max(S1.A_1)); celles de A'.A_2-A_1 sont (0, card-max(S2.A_2)); celles de A'.$A_1 \cap A_2$ sont (0, card-max(S1.A1)+card-max(S2.A2))

Mapping syntaxique :
$A' = S1.A_1,S2.A_2$; $A'.A_1\text{-}A_2 = S1.A_1,S2.A_2$; $A'.A_2\text{-}A_1 = S1.A_1,S2.A_2$; $A'.A_1 \cap A_2 = A_1,A_2$
$S1.A_1 = A'.A_1\text{-}A_2$, $A'.A_1 \cap A_2$; $S2.A_2 = A'.A_2\text{-}A_1$, $A'.A_1 \cap A_2$

Mapping des occurences:
$A' = S1.A_1 \cup S2.A_2$; $A'.A_1\text{-}A_2 = S1.A_1 - S2.A_2$; $A'.A_2\text{-}A_1 = S1.A_1 - S2.A_2$;
$A'.A_1 \cap A_2 = A_1$ si $A_1 = A_2$
$S1.A_1 = A'.A_1\text{-}A_2 \cup A'.A_1 \cap A_2$; $S2.A_2 = A'.A_2\text{-}A_1 \cup A'.A_1 \cap A_2$

$A_1 \neq A_2$, l'intégration fournit le résultat suivant:

Attribut composé de nom A' et de composants nommés A_1 et A_2.
Les cardinalités de A' sont (i,1)avec i=0 ou 1 suivant que les attributs A_1 et A_2 sont facultatifs ou non; Celles de $A'.A_1$ sont identiques à celles de A_1; celles de $A'.A_2$ sont identiques à celles de A_2.
Mapping syntaxique:$A' = S1.A_1,S2.A_2$; $A'.A_1=S1.A_1$; $A'.A_2 = S2.A_2$
S1.A1 = A'.A1 ; S2.A2 =A'.A2
Mapping des occurences: $A' = S1.A_1 \cup S2.A_2$; $A'.A_1=S1.A_1$; $A'.A_2 = S2.A_2$
S1.A1 = A'.A1 ; S2.A2 =A'.A2

Concernant les correspondances entre les attributs, on s'est limité à celles relatives à une assertion de correspondances d'équivalence entre les éléments parents. On peut toutefois étendre les solutions aux autres correspondances mais la prudence est de rigueur étant donné les possibilités de conflits entre correspondances des parents(TE...) et celles des fils(attributs). Soulignons l'exemple d'inclusion entre deux TE E1 et E2 et l'inclusion entre A2 et A1 appartenant respectivement à E2 et E1(E2 $\supset$ E1 et A1 $\supset$ A2)

[2]. Intégration d'éléments (TE, TA, attribut, lien) ne faisant pas partie d'une correspondance

Tout élément E n'intervenant pas dans une assertion de correspondance sera ajouté tel quel au schéma intégré.

Mapping syntaxique: E' ≡ E

Mapping des occurences: E' =E
E = E'

[3]. Intégration de deux TE et de leurs attributs respectifs

Soient E1 et E2 deux TE des schémas S1 et S2 respectivement et, $(A_{11},...,A_{1j},B_1,..,B_k)$ et $(A_{21},...,A_{2j},C_1,...,C_h)$ les attributs respectifs de E1 et E2.

Soit l'assertion de correspondance:

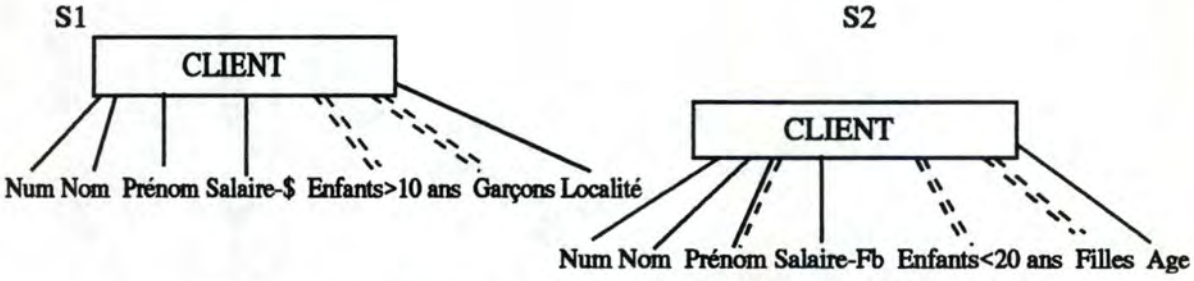
$$E1 \equiv E2 \text{ WCA } attcor_1(A_{11},A_{21}), attcor_2(A_{12},A_{22}),..., attcor_j(A_{1j},A_{2j})$$

L'intégration nous fournit un TE de nom E (peut faire l'objet d'une renomination) dont la structure est définie par l'union des attributs de E1 et E2 comme énoncé précédemment dans **(1)**.

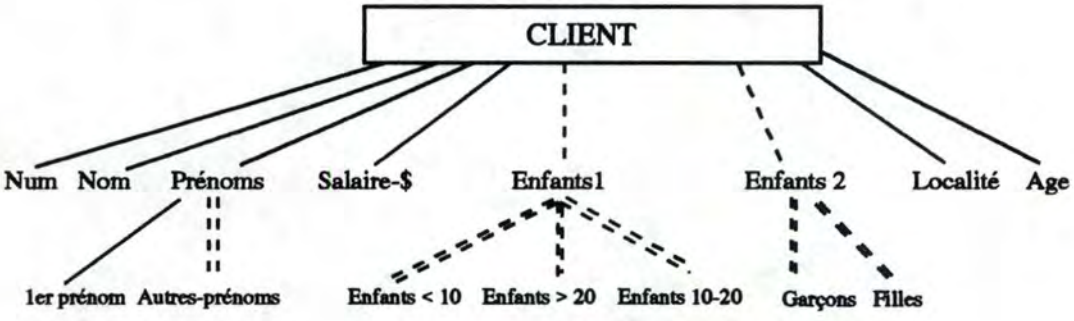
Mapping syntaxique: E ≡ E1 WCA cfr mapping de la règle [1]
$attcor(B'_1,B_1),...,attcor(B'_k,B_k)$ attcor est "="
E ≡ E2 WCA cfr mapping de la règle [1]
$attcor(C'_1,C_1),...,attcor(C'_h,C_h)$ attcor est "="

Mapping des occurences: E = Integrate-join (E1, E2, $attcor_1(A_{11},A_{21})$, $attcor_2(A_{12},A_{22})$,..., $attcor_j(A_{1j},A_{2j})$)

E1 =..., E2 =..La définition de ces mapping dépend des correspondances entre les attributs de ces éléments. L'exemple de la figure A.1 fournit un exmple de ce mapping, faisant intervenir les opérateurs de projection, de simplification, de fusion et de changement de nom.



S1.CLIENT ≡ S2.CLIENT WCA Num = Num, Nom=Nom, Prénom ⊂ Prénom, Salaire-$ = Fb-$(Salaire-Fb), Enfants>10 ans ∩ Enfants < 20ans, Garçons ≠ Filles



Mapping syntaxique :

CLIENT = S1.CLIENT WCA Num= Num, Nom=Nom, Prénoms. 1er prénom = Prénom, Salaire-$= Salaire-$, Enfants1.Enfants>20 ⊂ Enfants>10, Enfants1.Enfants 10-20 ⊂Enfants >10, Enfants2.Garçons= Garçons, Localité=Localité

CLIENT = S2.CLIENT WCA Num= Num, Nom=Nom, Prénoms = Prénom, Salaire-$= Fb-$(Salaire-Fb), Enfants1.Enfants<10 ⊂ Enfants<20, Enfants1.Enfants 10-20 ⊂ Enfants <20, Enfants2.Filles= Filles, Age= Age

Mapping des occurences :

CLIENT = Integrate-join(S1.CLIENT, S2.CLIENT, Num = Num, Nom=Nom, Prénom ⊂ Prénom, Salaire-$ = Fb-$(Salaire-Fb), Enfants>10 ans ∩ Enfants < 20ans, Garçons ≠ Filles

S1.CLIENT = Rename[Prénoms: Prénom, Enfants > 10 ans :Enfants1, Enfants2: Garçons] λ[Enfants1;Enfants > 20,Enfants 10-20] Σ[1er Prénom,Garçons] π [Num,Nom, Prénoms(1er Prénom), Salaire-$, Enfants1(Enfants>20,Enfants10-20), Enfants2(Garçons), Localité] CLIENT

S2.CLIENT = Rename[Prénoms: Prénom, Salaire-$:Salaire-Fb, Enfants <20 ans :Enfants1, Enfants2: Garçons] λ[Enfants1;Enfants <10,Enfants 10-20] Σ [Filles] π [Num,Nom, Prénoms, $-Fb(Salaire-$), Enfants1(Enfants<10,Enfants10-20), Enfants2(Filles), Age] CLIENT

Figure A.1: Exemple d'intégration

[4]. Intégration de deux liens

Cette règle traite de l'intégration de liens élémentaires(lien d'attributs, rôles) et permet l'intégration de deux liens équivalents qui lient des éléments équivalents.

Soient A1 et B1 deux éléments(TE, TA, attributs) liés dans le schéma S1, et A2 et B2 deux éléments(TE, TA, attributs) liés dans le schéma S2. Soit les correspondances suivantes:

$$A1 \equiv A2$$
$$B1 \equiv B2$$
$$A1 - B1 \equiv A2 - B2$$

Soit A l'élément du schéma intégré correspondant à A1 et A2,
Soit B l'élément du schéma intégré correspondant à B1 et B2,
Alors l'intégration des liens A1-B1 et A2-B2 est un lien A-B. Le type de lien dépend du type de A et B:
Si A ou/et B sont des attributs alors A-B est un lien d'attribut
Si A est un TE et B un TA (ou inversément) alors A-B est un rôle.
Les cardinalités du lien A-B est identique à celle de A1-B1 et A2-B2 car il s'agit de correspondances d'équivalence.

Mapping syntaxique : A-B ≡ A1-B1
A-B ≡ A2-B2

Mapping des occurences : A-B = rename[A1-B1]
A-B = rename[A2-B2]
A1-B2 = rename[A-B]
A2-B2 = rename[A-B]

Des règles [2] et [4], on déduit que l'intégration de deux TA (TA1 et TA2) équivalent est un TA qui lient tous les TE que TA1 et TA2 lient.**[5]**

De même, l'intégration d'un TE E1 et d'un TA TA équivalents est un TE E et deux TA TA1 et TA2 liant E et les TE que liaient TA.**[6]**

[7]. Intégration de deux chemins

Un chemin E_1-E_2-...-E_{p-1}-E_p est porteur d'information si

-il existe un des E_i i:2..p-1 qui possède au moins un attribut
- ou/et il existe au moins deux X_i :i :1...p-1 dont la cardinalité maximale dans le lien E_i-E_{i+1} est supérieure à 1.

Un chemin ne vérifiant pas une de ces caractéristiques est un lien direct entre E_1 et E_p.

Soient $E_1,E_2..,E_n$ les éléments du schéma S1,
Soient $F_1,F_2,...,F_p$ les éléments du schéma S2,
avec les correspondances suivantes: $E_1 \equiv F_1$, $E_n \equiv F_p$
Soient G_1(respectivement Gn) l'élément du schéma intégré correspondant à E_1 et F_1(à E_n et F_p respectivement),
alors la correspondance entre les chemins E_1-E2-...-E_n $\equiv$ F_1-F_2-...-F_p génère dans le schéma intégré :

- si E1-E_2-...-E_n n'est pas un chemin porteur d'information (ou inversément l'autre): **l'autre chemin G_1-F'_2-.....-F'_{p-1}-G_p** où $F'_2,...,F'_{p-1}$ sont les éléments du schéma intégré correspondants à $F_2,...,F_{p-1}$ et où chaque lien composant de G_1-F'_2-...-F'_{p-1}-G_p est créé en accord avec les concepts de modélisation des éléments liés, comme dans la règle [4]

- si les deux chemins sont porteurs d'information:
les deux chemins et une contrainte d'intégrité; les chemins sont G_1-E'_2-.....-E'_{n-1}-G_p et G_1-F'_2-.....-F'_{p-1}-G_p où $E'_2,...,E'_{n-1}$ sont les éléments du schéma intégré correspondants à $E_2,...,E_{n-1}$ et $F'_2,...,F'_{p-1}$ sont les éléments du schéma intégré correspondants à $F_2,...,F_{p-1}$ et où chaque lien composant des chemins est créé en accord avec les concepts de modélisation des éléments liés, comme dans la règle [4]. La contrainte d'intégrité spécifie que les deux chemins lient les mêmes occurences.

[8] Intégration d'un TE avec un attribut

Soient E1 (schéma S1)un TE avec ses attributs ($A_{11},...,A_{1j},B_1,..,B_k$) et E2 un attribut composé du TE X2 (schéma S2) avec les attributs composants ($A_{21},...,A_{2j},C_1,...,C_h$) **ou** un attribut atomique. Dans ce dernier cas, on considère que E2 a lui-même comme composant.

Soit la correspondance :

$$E1 \equiv E2 \text{ WCA } attcor_1(A_{11},A_{21}),\ attcor_2(A_{12},A_{22}),...,\ attcor_j(A_{1j},A_{2j})$$

Soit X le TE correspondant à X2 dans le schéma intégré; les éléments dans le schéma intégré résultant de l'intégration de E1 et E2 sont un TE E avec un TA binaire entre X et E tel que :

- l'attribut E2 de X2 est transformé en rôle référencant E; ses connectivités sont égales aux cardinalités de E2
- le nom de E est arbitrairement choisi
- la structure de E consiste en l'union des attributs de E1 et E2, comme défini par l'opérateur integrate-join

Mapping Syntaxique : E $\equiv$ E1 WCA cfr mapping de la règle [1]
attcor(B'_1,B_1),...,attcor(B'_k,B_k) attcor est "="
E $\equiv$ E2 WCA cfr mapping de la règle [1]
attcor(C'_1,C_1),...,attcor(C'_h,C_h) attcor est "="

X-E $\equiv$ X2-E2

Mapping des occurences : E= Integrate-join(E1,E2, $attcor_1(A_{11},A_{21})$, $attcor_2(A_{12},A_{22})$,...,
$attcor_j(A_{1j},A_{2j})$)

X-E = rename[X2-E2]

[9] Intégration de deux TE en correspondance d'inclusion

Soient $E1(A_{11},...,A_{1j},B_1,..,B_k)$ TE du schéma S1 et $E2(A_{21},...,A_{2j},C_1,...,C_h)$ TE du schéma S2 avec la correspondance suivante :

$$E1 \supset E2 \text{ WCA } A_{11} \supset A_{21},..,A_{1j} \supset A_{2j}$$

On produira le schéma intégré suivant :

$E'1(A'_1,A'_2,..,A'_j,B'_1,..,B'_k)$
$E'2(C'_1,...,C'_h)$
$E'2$ is-A $E'1$

Mapping : $E'1 = E1$ WCA $(A'_1=A_{11},A'_2= A_{12}, \ldots , A'_j= A_{1j}, B'_1=B_1, \ldots ,B'_k=B_k)$
$E'2 = E2$ WCA $(C'_1=C_1, \ldots ,C'_h=C_h)$
$E1= E'1$ WCA $(A_1=A'_1,A_1=A'_2,..,A_j=A'_j,B_1=B'_1,..,B_k=B'_k)M$
$E2= E'1$ WCA $(A_1=A'_1,A_2=A'_2,..,A_j=A'_j)$, $E'2$ WCA$(C_1=C'_1,...,C_h=C'_h)$

Mapping des occurences : $E'1 = E1$
$E'2 = \pi[C_1,...,C_h] E'2$
$E1=$ rename[....] $E'1$
$E2=$ rename[....] $E'2 \Delta E'1$

Nous avons ainsi proposé un échantillon de règles d'intégration. Cette liste devrait être étendue par les autres types de correspondances(Intersection, disjonction) en tenant compte des combinaisons de correspondances entre TE/TA et leurs attributs respectifs.De nouveaux opérateurs algébriques devrons sans aucun doute être définis afin de pourvoir aux règles de mapping issues de ces nouvelles règles d'intégration.